Bruno Garbe Junior

Análise de Algoritmos de Roteamento Baseados em Formigas

Dissertação apresentada a Escola Politécnica da
Universidade de São Paulo para a obtenção do
Título de Mestre em Engenharia.

São Paulo

2006

Bruno Garbe Junior

Análise de Algoritmos de Roteamento Baseados em Formigas

Dissertação apresentada a Escola Politécnica da
Universidade de São Paulo para a obtenção do
Título de Mestre em Engenharia.

Área de Concentração:
Engenharia de Sistemas

Orientador:
José Jaime da Cruz

São Paulo
2006

Deus,

dai-me a serenidade para aceitar as coisas

que eu não posso mudar,

coragem para mudar as coisas que eu possa,

e sabedoria para que eu saiba a diferença.

Dedicatória

Aos meus pais e meu irmão. Só posso dizer o quão feliz o sou por tê-los como minha família, e o quão pouco eu seria sem vocês.

À minha namorada, amante e futura esposa, Tati. Se todas as outras pessoas me ajudaram e me deram condição de viver uma vida, foi você que me deu o motivo de vivê-la.

Agradecimentos

Ao orientador e amigo Jaime. Tive o orgulho e privilégio de tê-lo como orientador nesse trabalho, e durante todo o caminho ele foi muito mais do que isso, foi um grande amigo e companheiro.

Ao CNPq pelo auxílio na forma de bolsa.

À todos os demais que direta ou indiretamente colaboraram para a elaboração deste trabalho.

Resumo

Roteamento por colônia de formigas é um método de roteamento em redes de comunicação, e diversos algoritmos foram propostos nos últimos anos baseado nessa estrutura. Todos esses algoritmos produzem excelentes resultados, provando a sua eficiência e eficácia. Este trabalho apresenta os resultados de desempenho dos principais algoritmos encontrados na literatura, e com base nesses resultados, propõe um novo algoritmo com desempenho equivalente e com uma complexidade computacional menor. O trabalho é focalizado em redes tipo datagrama com topologia irregular, descrevendo suas propriedades e características e realizando uma análise e comparação de seus desempenhos em um ambiente de simulação.

Abstract

Ant Colony Routing is an adaptive method for routing in communication networks, and several algorithms have been proposed in the last years based on this framework. All these algorithms show excellent results, proving their efficiency and efficacy. This work presents the results of the performance of the main algorithms found in the literature, and based on these results, it proposes a novel algorithm that has a similar performance but with a lower computational complexity. The work is focused in datagram like networks with irregular topology, describing its characteristics and properties. The performances in an simulation environment are analysed and compared.

Sumário Resumido

\	Lista de Figuras	
\	Lista de Tabelas	
\	Lista de Abreviaturas e Siglas	
1 \	Introdução	■ 1
2 \	Algoritmos de Roteamento	■ 4
3 \	Roteamento Baseado em Agentes	■ 11
4 \	AntNet	■ 20
5 \	Trail Blazer	■ 37
6 \	Modelos e Implementação	■ 47
7 \	Resultados Obtidos e Discussão	■ 60
8 \	Conclusões	■ 87
A \	Resultados Completos das Simulações	■ 89
\	Referências	■ 112

Sumário

\ **Lista de Figuras**

\ **Lista de Tabelas**

\ **Lista de Abreviaturas e Siglas**

1 \ Introdução ■ 1

1.1 Objetivos □ 1

1.2 Motivação □ 1

1.3 Nota sobre os Termos em Inglês □ 3

2 \ Algoritmos de Roteamento ■ 4

2.1 Roteamento em Redes □ 4

2.2 Classificação dos Algoritmos □ 5

2.3 Principais Algoritmos de Roteamento □ 7

2.3.1 Inundação · 7

2.3.2 Roteamento pelo caminho mais curto · 7

2.3.3 Roteamento por Vetor de Distância · 7

2.3.4 Roteamento por Estado de Enlace · 9

3 \ Roteamento Baseado em Agentes ■ 11

3.1 Resolução de Problemas Coletivamente □ 11

3.2 Desenvolvimento de Agentes Artificiais □ 12

3.2.1 Inteligência de Enxames · 14

3.2.2 Otimização por Colônia de Formigas □ 15

3.2.3 Conceito de Estigmergia · 15

3.3 Isomorfismo entre Agentes Biologicos e Agentes Artificiais □ 16

3.3.1 Agentes Inteligentes · 17

3.3.2 Tabelas de Roteamento e Tabelas de Feromônio · 17

3.3.3 Calculo das Probabilidades · 18

3.4 Algoritmos Propostos na Literatura □ 19

4 \ AntNet ■ 20

4.1 Algoritmo AntNet □ 20

4.1.1 Estruturas de Dados · 21

4.1.2 Descrição dos Agentes · 24

4.1.2.1 Criação dos Agentes · 25

4.1.2.2 Seleção dos Destinos · 25

4.1.3 Roteamento dos Agentes em Avanço · 25

4.1.4 Roteamento dos Agentes em Retorno · 27

4.1.5 Roteamento dos Pacotes de Dados · 27

4.1.6 Atualização das Estruturas de Dados · 28

4.1.6.1	Atualização do Modelo de Tráfego Local	· 29
4.1.6.2	Atualização da Tabela de Roteamento	· 29
4.1.6.3	Cálculo do Reforço	· 30
4.1.7	Resumo dos Parâmetros	· 32
4.2	Algoritmo AntNet-FA	□ 33

5 \ Trail Blazer ■ 37

5.1	Algoritmo Trail Blazer	□ 38
5.1.1	Estruturas de Dados	· 39
5.1.1.1	Inicialização da Tabela de Roteamento	· 39
5.1.2	Descrição de Agentes	· 40
5.1.3	Gerenciando Pacotes em Vértices Intermediários	· 40
5.1.3	Roteamento dos Pacotes de Dados	· 43
5.1.5	Atualizando a Tabela de Roteamento	· 44
5.1.6	Resumo dos Parâmetros	· 45
5.2	Algoritmo Trail Blazer Simple	□ 45
5.3	Algoritmo Trail Blazer Uniform	□ 46

6 \ Modelos e Implementação ■ 47

6.1	Modelo da Rede de Comunicações	□ 47
6.1.1	Topologia	· 47
6.1.2	Nós e Enlaces	· 48
6.1.3	Padrões de Tráfego	· 49
6.2	Simulação de Redes de Comunicação	□ 49
6.2.1	Sistemas de Eventos Discretos	· 49

6.2.2	OMNeT++	· 49
6.2.3	Módulos Hierárquicos	· 50
6.2.4	Mensagens, Portas e Enlaces	· 50
6.2.5	Transmissão de Mensagens	· 51
6.3	Implementação do Simulador	□ 51
6.3.1	Estrutura Geral do Simulador	· 52
6.3.2	Pacotes de Dados e de Agentes	· 52
6.3.3	Gerador de Agentes	· 54
6.3.4	Sorvedouro de Agentes	· 55
6.3.5	Ninho dos Agentes	· 55
6.3.6	Gerador de Dados	· 56
6.3.7	Sorvedouro de Dados	· 56
6.3.8	Fila do Roteador	· 56
6.3.9	Tabela de Roteamento	· 58
6.3.10	Roteador Interno	· 59

7 \ Resultados Obtidos e Discussão ■ 60

7.1	Porcentagem dos Pacotes Entregues	□ 61
7.2	Atraso dos Pacotes	□ 66
7.3	Largura de Banda Utilizada	□ 73
7.4	Número de Saltos dos Pacotes de Dados	□ 77
7.5	Número de Agentes Gerados e Coletados	□ 79
7.6	Tempo de Latência dos Agentes	□ 79

7.7 Largura de Banda Utilizada pelos Agentes □ 79

7.8 Sensibilidade dos Algoritmos em Relação aos Agentes □ 85

8 \ Conclusões ■ 87

A \ Resultados Completos das Simulações ■ 89

\ Referências ■ 112

Lista de Figuras

- Figura 3.1 Demonstração da capacidade das formigas de encontrar um caminho ótimo do formigueiro até a fonte de comida. (a) inicialmente não existe nenhum feromônio no caminho e as formigas escolhem aleatoriamente o caminho a seguir. (b) como o caminho B é mais curto, essa formiga retorna primeiro para o formigueiro. (c) a terceira formiga escolhe o caminho com base na intensidade de feromônio, escolhendo a rota B. (d) a maioria das formigas acaba utilizando o caminho mais curto. Retirado de [24]. · 13
- Figura 3.2 Caso o caminho B seja interrompido, as formigas conseguem explorar o ambiente para encontrar outros caminhos de menor distância até a fonte de comida. Retirado de [24]. · 14
- Figura 4.1 Estruturas de dados utilizadas pelos agentes no AntNet para o caso do nó com L vizinhos e a rede com N nós. A tabela de roteamento (feromônio) é organizada como em algoritmos de vetor de distância, mas as entradas não são distâncias mas probabilidades indicando a “qualidade” relativa dos enlaces possíveis para o próximo salto. A tabela de roteamento de dados é obtida a partir da transformação da tabela de feromônio. O modelo paramétrico armazena as informações relativas da distribuição do tráfego, para o cálculo da qualidade relativa dos diferentes caminhos. O estado atual das filas dos enlaces também é utilizado pelo AntNet. Retirado de [3] · 22
- Figura 4.2 Agentes em avanço no AntNet e agentes em avanço no AntNet-FA. Os agentes em avanço no AntNet-FA utilizam uma fila de alta prioridade nos roteadores atualizando as tabelas de roteamento. · 35
- Figura 6.1 Topologia da rede japonesa NTT. Cada arco representa um enlace bi-direcional. A largura de banda é de 6Mbits/sec. Atrasos de propagação variam entre 1 e 5ms. · 48
- Figura 6.2 Dois submódulos do OMNeT demonstrando como eles se interligam hierarquicamente. (a) Dois submódulos conectados um ao outro. (b) dois submódulos conectados ao módulo pai. · 50
- Figura 6.3 Estrutura Básica dos módulos do simulador. Cada caixa cinza representa um módulo separado do simulador. As linhas com setas entre os módulos repre-

	sentam linhas de comunicação entre os módulos (não representam necessariamente enlaces, apesar de enlaces serem representados por elas). . 53
Figura 6.4	Descrição do módulo de mensagem de dados . 54
Figura 6.5	Descrição do módulo de mensagem dos agente . 54
Figura 6.6	Descrição do módulo de geração dos agentes. . 55
Figura 6.7	Descrição do módulo de ninho dos agentes do AN e ANFA. . 56
Figura 6.8	Descrição do módulo de ninho dos agentes do TB, TBs e TBu. . 57
Figura 6.9	Descrição do módulo de roteador interno. . 58
Figura 7.1	Resultados para a porcentagem de pacotes entregues para os diferentes algoritmos de roteamento. . 63
Figura 7.2	Dados obtidos para a porcentagem de pacotes entregues em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos. . 64
Figura 7.3	Média do atraso dos pacotes de dados, em segundos. . 68
Figura 7.4	Desvio Padrão do atraso dos pacotes de dados, em segundos. . 69
Figura 7.5	Intervalo máximo de atraso para 90% dos pacotes, em segundos. . 70
Figura 7.6	Resultados obtidos para o intervalo máximo do atraso de 90% dos pacotes em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos. . 71
Figura 7.7	Largura de banda utilizada pelos diferentes algoritmos. . 74
Figura 7.8	Largura de banda utilizada em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos. . 75
Figura 7.9	Número de saltos médios para os pacotes de dados para os diferentes algoritmos de roteamento. . 78
Figura 7.10	Número de agentes gerados (10^5) pelos diferentes algoritmos. . 80
Figura 7.11	Porcentagem de agentes utilizados no reforço que voltaram a sua fonte. . 81
Figura 7.12	Tempo de latência máximo para 90% dos agentes, em segundos. . 82
Figura 7.13	Número médio de saltos dos agentes dos diferentes algoritmos. . 83
Figura 7.14	Largura de banda utilizada pelos agentes em relação à largura de banda total da rede, em 10^{-5} . . 84
Figura 7.15	Porcentagem de pacotes de dados entregue em função do número de agentes lançados para os diferentes algoritmos. . 86
Figura 7.16	Intervalo máximo de atraso para 90% dos pacotes de dados em função do número de agentes lançados para os diferentes algoritmos. . 86

Lista de Tabelas

Tabela 3.1	Analogias entre agentes biológicos e agentes artificiais.	· 16
Tabela 3.2	Algoritmos propostos na literatura para o problema de roteamento em redes de comunicação utilizando agentes, tanto para o roteamento em redes orientadas a conexão quanto para redes não-orientadas a conexão.	· 19
Tabela 4.1	Estrutura dos Agentes do AntNet. Tanto os agentes em avanço quanto os agentes em retorno utilizam a mesma estrutura básica. Eles simplesmente percebem diferentes entradas e saídas, tendo o seu comportamento diferenciado na rede por essas diferenças.	· 24
Tabela 4.2	Sumário dos parâmetros utilizados no AntNet.	· 33
Tabela 5.1	Estrutura dos Agentes do Trail Blazer. Assim com no AntNet, tanto os agentes em avanço quanto os agentes em retorno utilizam a mesma estrutura básica, mas ao contrário do AntNet existem três tipos de agentes: os agentes exploradores, os agentes de melhor rota e os agentes em retorno.	· 41
Tabela 5.2	Sumário dos parâmetros utilizados no Trail Blazer.	· 45
Tabela 7.1	Resultados para a porcentagem de pacotes entregues para os diferentes algoritmos de roteamento.	· 63
Tabela 7.2	Média do atraso pacotes de dados, em segundos.	· 68
Tabela 7.3	Desvio Padrão do atraso dos pacotes de dados, em segundos.	· 69
Tabela 7.4	Intervalo máximo de atraso para 90% dos pacotes, em segundos.	· 70
Tabela 7.5	Largura de banda utilizada pelos diferentes algoritmos.	· 74
Tabela 7.6	Número de saltos médios para os pacotes de dados para os diferentes algoritmos de roteamento.	· 78
Tabela 7.7	Número de agentes gerados (10^5) pelos diferentes algoritmos.	· 80
Tabela 7.8	Porcentagem de agentes utilizados no reforço que voltaram a sua fonte.	· 81
Tabela 7.9	Tempo de latência máximo para 90% dos agentes, em segundos.	· 82
Tabela 7.10	Número médio de saltos dos agentes dos diferentes algoritmos.	· 83

Tabela 7.11	Largura de banda utilizada pelos agentes em relação à largura de banda total da rede, em 10^{-5} .	· 84
Tabela A.1	Resultados experimentais obtidos para o algoritmo OSPF.	· 90
Tabela A.2	Resultados experimentais obtidos para o algoritmo AN.	· 92
Tabela A.3	Resultados experimentais obtidos para o algoritmo ANFA.	· 96
Tabela A.4	Resultados experimentais obtidos para o algoritmo TB.	· 100
Tabela A.5	Resultados experimentais obtidos para o algoritmo TBs.	· 104
Tabela A.6	Resultados experimentais obtidos para o algoritmo TBu.	· 108

Lista de Abreviaturas e Siglas

AN	AntNet
ANFA	AntNet Flying Ants
FDDI	Fiber Distributed Data Interface
FIFO	First in, First out
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
LSA	Link State Advertisements
NTT	Nippon Telephone and Telegraph Company
OSI	Open System Interconnection
OSPF	Open Shortest Path First Protocol
QoS	Quality of Service
RIP	Routing Information Protocol
STL	Standard Template Library
TB	Trail Blazer
TBs	Trail Blazer simple
TBu	Trail Blazer uniform
TCP	Transfer Control Protocol

Capítulo 1

Introdução

1.1 Objetivos

O presente trabalho tem como finalidade apresentar algoritmos baseados no comportamento de formigas para resolver o problema de encaminhamento de dados para aliviar o congestionamento nas redes e verificar a melhora dos resultados no problema de roteamento adaptativo utilizado em diferentes sistemas de comunicação de dados. O trabalho é focalizado em redes tipo datagrama com topologia irregular, descrevendo suas propriedades e características e realizando uma análise e comparação de seus desempenhos em um ambiente de simulação.

1.2 Motivação

A necessidade da troca de informações entre pessoas do mundo todo tem gerado uma demanda mundial por serviços de comunicação que cresce exponencialmente. Nesse sentido temos como exemplo maior a Internet, que em 2004 possuía conexões em mais de 100 países e mais de 10 milhões de usuários [2]. Dessa forma, as redes de comunicação modernas tornaram-se normalmente grandes, complexas e dinâmicas. Essa natureza dinâmica das redes é um resultado das demandas de tráfego e mudanças nos parâmetros da rede que são normalmente impossíveis de se prever, o que dificulta a sua configuração e manutenção para obter resultados ótimos.

A natureza dinâmica e normalmente imprevisível das redes de comunicação criou a necessidade de algoritmos de roteamento que sejam capazes de se adaptar autonomicamente a mudanças nas demandas de tráfego e condições da rede. O aumento do tamanho e complexidade das redes e a atual necessidade crescente pela intercomunicação de dados tem motivado uma grande pesquisa de novos algoritmos de roteamento adaptativos, baseados nos recentes avanços de diversas áreas, visando tornar possível a comunicação em diferentes redes e topologias.

Uma das propostas mais promissoras tem sido uma nova classe de algoritmos que tiveram o seu desenvolvimento inspirado no comportamento das colônias de formigas ao determinar o tradicional problema do caminho mais curto^{1.1} entre dois pontos. A intenção é reproduzir, em um sistema artificial, algumas das habilidades de resolver problemas coletivamente apresentadas pelos agentes biológicos.

Essa classe utiliza agentes (formigas) que autonomicamente atravessam a rede e coletivamente constroem uma ação distribuída de roteamento. Algoritmos de roteamento baseados em agentes possuem as seguintes propriedades [1]:

1. Não requerem um modelo completo da rede, isto é, um conhecimento *a priori* dos custos dos enlaces, capacidades ou demandas do tráfego.
2. Utilizam uma estrutura de controle descentralizada, assim as operações do algoritmo não dependem da integridade de um único controlador central.
3. Eles possuem a capacidade de se adaptar autonomicamente às mudanças nos parâmetros da rede e demandas de tráfego.

Percebe-se facilmente que os algoritmos de roteamento baseados em agentes não são a única classe de algoritmos que possuem essas propriedades descritas anteriormente, entretanto, a obtenção dessas propriedades através do uso de agentes é única dessa classe, e constitui uma importante mudança de paradigma. Tradicionalmente, os algoritmos propostos na literatura utilizam uma abordagem passiva no monitoramento e obtenção no problema de adaptação do tráfego. Ao contrário, os algoritmos baseados em agentes utilizam *experimentos aleatórios controlados* (as formigas) para obter ativamente informações não-locais úteis sobre as características da solução [3]. Essa informação é utilizada continuamente para atualizar as políticas de roteamento nos nós de decisão, através das tabelas de feromônios.

1.1 Do inglês: Shortest-Path

1.3 Nota sobre os Termos em Inglês

Em vista do grande número de termos em inglês que não possuem uma tradução padronizada em português e tentando-se evitar uma multiplicação de termos traduzidos, esses termos são apresentados no texto em português, sendo que na primeira vez que um termo que é tradicional em inglês é utilizado, é apresentada a sua grafia original, na forma de uma nota de rodapé. Em alguns casos somente, na qual não existe uma tradução com o mesmo sentido em português, ou o termo em inglês é muito tradicional, manteve-se o termo em inglês.

Capítulo 2

Algoritmos de Roteamento

2.1 Roteamento em Redes

Uma rede de comunicação consiste de um conjunto de nós que são interligados por um conjunto de enlaces. Sobre esses dois conjuntos é colocada uma demanda de tráfego que permite a comunicação e troca de informações entre os nós. Para que esse tráfego alcance o seu destino ele precisa ser aceito pelos mecanismos de admissão e fluxo da rede, que verificam a possibilidade de se entregar esse tráfego de informações. Uma vez que o tráfego seja aceito pelos mecanismos de admissão e fluxo, é tarefa do mecanismo de roteamento determinar a rota, ou rotas, na qual uma unidade de comunicação (mensagem ou pacote) é movida através de um caminho de um nó fonte até um nó destino.

Dessa forma, pode-se definir roteamento [15] como o processo de escolha de um determinado caminho através de uma rede sobre a qual se transmitem pacotes de um determinado nó fonte até um determinado nó destino. Na maioria dos casos, esses pacotes precisam atravessar diversos outros nós para realizar a sua transmissão e podem sofrer atrasos em cada um dos nós ou enlaces que eles percorrem. O atraso que afeta um pacote em cada enlace é a soma do tempo de armazenamento, processamento e atraso de transmissão [1]. O tempo total de transmissão experimentado por um pacote viajando entre um determinado par de nós fonte-destino é a soma de todos os atrasos dos enlaces utilizados.

Existem diversas maneiras de se determinar esse caminho (ou caminhos) em uma rede de co-

municações. Uma das formas mais simples é utilizando-se da teoria de grafos e modelando a rede como um grafo para se aplicar algum dos algoritmos padrões, como o de Dijkstra ou do fluxo máximo, para se encontrar o melhor caminho. Para redes que variam esporadicamente essa solução é válida. Infelizmente, para a grande maioria das redes globais, essa abordagem não é ótima, pois nesse caso as redes estão sempre mudando suas características, sendo necessário atualizar as rotas de melhor caminho. Outro problema aparece quando se considera que a abordagem anterior é muito centralizada, apresentando o problema de um processamento maior com o aumento do tamanho da rede, gerando questões como desempenho, disponibilidade e segurança.

Uma solução possível para esse problema é determinar algoritmos distribuídos, implementados em cada um dos vértices da rede, que tenham como objetivo melhorar o desempenho da rede como um todo a partir de decisões e medidas locais. Dentro desse paradigma, pode-se novamente definir [16] que o algoritmo de roteamento é a parte da camada de rede responsável por decidir em cada nó do caminho qual linha de saída um determinado pacote que está chegando deve ser transmitido. Portanto, um algoritmo de roteamento é um procedimento matemático que um roteador utiliza para calcular entradas para sua tabela de roteamento, que é uma tabela convencional com linhas e colunas que pertence a um vértice da rede e é utilizada para armazenar as informações de roteamento. Esses algoritmos que definem a rota a ser seguida e as estruturas de dados que eles utilizam são de grande importância para a área de projeto de camadas de redes. Pode-se perceber, a partir dessas definições, que o objetivo de todo algoritmo de roteamento é direcionar o tráfego das fontes para os destinos maximizando a performance da rede.

Dessa forma, o problema generalizado de se determinar um algoritmo de roteamento ótimo pode ser enunciado como um problema de otimização multiobjetivo em um ambiente estocástico não-estacionário [7].

2.2 Classificação dos Algoritmos

Tradicionalmente os algoritmos de roteamento podem ser classificados de acordo com os seguintes critérios [16]:

- Algoritmos Estáticos ou Adaptativos.
- Algoritmos Centralizados ou Distribuídos.

- Algoritmos Mínimos ou Multi-caminhos.

Algoritmos estaticos ou *não-adaptativos* não baseiam suas decisões em métricas e/ou estimativas do tráfego atual e topologia. Ao invés disso, a escolha do caminho é determinada anteriormente à necessidade de envio da mensagem, e distribuída para todos os roteadores. Esse procedimento é normalmente chamado de roteamento estático.

Algoritmos adaptativos, em contraste, modificam suas rotas para refletir mudanças na topologia e tráfego. Os algoritmos adaptativos diferenciam-se entre si por onde eles conseguem suas informações (ex.: localmente, roteadores adjacentes, etc.), quando ocorre a troca de rotas e qual métrica é utilizada para otimização.

Outra classificação possível diz respeito ao local onde é realizado o processamento das informações: caso seja um local central, que posteriormente distribui as informações para todos os vértices, é um *algoritmo centralizado*. Caso cada vértice calcule sua tabela de roteamento, tem-se um *algoritmo distribuído*.

Por [7] sabe-se ainda que roteadores podem ser divididos em duas categorias: *de caminho mínimo* e *multi-caminhos* (ou *de caminho não-mínimo*). Roteadores de caminho mínimo permitem que os pacotes escolham somente caminhos de custo mínimo, e normalmente utilizam roteamento do caminho mais curto. Em contraste, os roteadores de multi-caminho permitem uma escolha entre todos os caminhos possíveis, seguindo alguma estratégia heurística. Dessa forma, eles conseguem "balancear" o tráfego sobre múltiplas rotas. O propósito disso é evitar a sobrecarga de enlaces individuais ou rotas com excessiva quantidade de tráfego. Isso é normalmente chamado de *balanceamento de carga*.

Por último, as redes ainda podem ser classificadas como *orientadas a conexão* ou *não-orientadas a conexão*. Em redes não-orientadas a conexão, cada pacote de dados pertencente a um determinado fluxo de dados é roteado como uma entidade independente. Em contraste, em redes orientadas a conexão, alguma forma de comunicação bi-direcional entre os nós de origem e de destino é estabelecida durante a fase de "inicialização" anterior à transmissão dos dados.

2.3 Principais Classes de Algoritmos de Roteamento

2.3.1 Inundação

O algoritmo de inundação^{2.1} é a abordagem mais simples. Corresponde a um algoritmo estático, no qual todo pacote que está para chegar é replicado em todas as linhas de saída exceto a linha pela qual chegou. Percebe-se facilmente que a inundação gera um número exponencialmente crescente de pacotes, a não ser que exista alguma medida para parar o processo. Normalmente se utiliza um contador nos pacotes que determina sua vida útil.

2.3.2 Roteamento pelo caminho mais curto

O roteamento pelo caminho mais curto^{2.2} possui uma perspectiva fonte-destino. Seu objetivo é determinar o menor caminho entre dois vértices, onde o custo de enlace é calculado seguindo-se alguma descrição física ou/e estatística do estado do enlace.

Por [7], sabe-se que se o custo do enlace for estático e refletir tanto a capacidade de transmissão do enlace como a medida do tráfego esperado, então o cálculo do caminho mais curto será ótimo na média para situações de tráfego estacionário. Caso as condições de tráfego sejam não-estacionárias, tem-se uma grande perda de eficiência.

Caso os custos sejam calculados de alguma maneira dinâmica, baseados em alguma medida do estado de congestionamento do enlace, um forte efeito de realimentação é introduzido entre a política de roteamento e os padrões de tráfego. Isso acaba levando a oscilações indesejáveis no sistema.

2.3.3 Roteamento por Vetor de Distância

Os algoritmos por vetor de distância^{2.3} (também conhecidos como algoritmos de Bellman-Ford) pertencem à classe de algoritmos pelo caminho mais curto e, portanto, utilizam métricas conhecidas como *custos* para determinar o melhor caminho para um determinado destino. O caminho com menor custo total é escolhido como o melhor caminho. No roteamento por vetor de distância, cada roteador mantém uma tabela de roteamento (vetor) consistindo de um

2.1 Do inglês: "Flooding"

2.2 Do inglês: "Shortest Path Routing"

2.3 Do inglês: "Distance Vector Algorithm"

conjunto de triplas da forma (Destino, Distância Estimada, Próximo Salto), definida para todos os destinos na sub-rede. A métrica utilizada pode ser o número de saltos, o tempo de atraso em milissegundos, o número de pacotes na fila através do caminho, ou algo similar. Essas tabelas são atualizadas trocando-se informações com seus vizinhos.

Um algoritmo por vetor de distância, simplificado, funciona da seguinte maneira:

1. Inicialmente, o roteador monta uma lista de quais roteadores ele pode alcançar, e qual a sua distância (normalmente em número de saltos). Em princípio isso significa medir a distância dos seus vizinhos, que se encontram a somente 1 salto de distância. Essa tabela é a tabela de roteamento.
2. Periodicamente a tabela de roteamento é compartilhada com outros roteadores utilizando-se algum protocolo específico entre roteadores. Esta informação é compartilhada somente entre roteadores fisicamente conectados (vizinhos), então, roteadores mais distantes ainda não recebem as novas tabelas.
3. Uma nova tabela de roteamento é construída baseada nas informações das interfaces de rede (passo 1), com a adição das novas informações recebidas de outros roteadores.
4. Caminhos de rede ruins são eliminados da nova tabela. Se dois caminhos idênticos para o mesmo roteador existem, somente o com menor métrica é mantido.
5. A nova tabela é então comunicada para todos os vizinhos do roteador. Dessa forma a informação de roteamento se espalha e eventualmente todos os roteadores saberão o caminho de roteamento para cada destino, qual roteador utilizar para se atingir esse destino, e qual o roteador para o próximo salto.

Os protocolos por vetor de distância são simples e eficientes em pequenas redes, e necessitam de pouca manutenção. Entretanto, eles não são muito escalonáveis, tendo uma propriedade de convergência pobre, o que levou ao desenvolvimento de algoritmos mais complexos e mais escalonáveis para grandes redes.

Apesar disso, os protocolos de vetor de distância ainda estão em uso atualmente: como exemplo tem-se o RIP (Routing Information Protocol) versão 1 e versão 2, e o IGRP (Interior Gateway Routing Protocol) desenvolvido pela Cisco.

2.3.4 Roteamento por Estado de Enlace

Algoritmos de estado de enlace^{2.4} trabalham com a mesma estrutura básica dos algoritmos de vetor de distância ao considerar que ambos favorecem a escolha do caminho de menor custo. Entretanto, protocolos de estado de enlace trabalham de uma maneira mais localizada. Eles enviam a informação de roteamento de suas tabelas para todos os vértices na sub-rede. Cada roteador, entretanto, envia somente a informação relativa ao estado dos seus próprios enlaces.

Dessa forma, a idéia do roteamento por estado de enlace é simples e pode ser resumida em cinco partes [16]:

1. Descobrir seus vizinhos e aprender seus endereços de rede;
2. Medir o atraso ou custo para cada vizinho;
3. Construir um pacote contendo as informações descobertas;
4. Mandar esse pacote para todos os outros roteadores;
5. Calcular o menor caminho para cada roteador.

Este processo é melhor para grandes ambientes que podem sofrer mudanças mais frequentemente. Algoritmos de estado de enlace permitem que os roteadores se focalizem nos seus próprios enlaces e interfaces. Qualquer roteador em uma rede só terá conhecimento direto dos roteadores e redes que diretamente se interligam a ele (ou seja, o estado dos seus próprios enlaces). Em um grande ambiente, isso significa que o roteador utiliza menos processamento para calcular caminhos mais complicados.

Outra vantagem desse processo localizado é que os protocolos podem manter tabelas de roteamento menores. Como o protocolo de estado de enlace somente mantém informação de roteamento para suas interfaces diretas, a tabela de roteamento contém muito menos informação que a tabela do protocolo por vetor de distância, que possui informação para múltiplos roteadores.

Assim como os protocolos por vetor de distância, protocolos de estado de enlace necessitam de atualizações para compartilhar informações de um com outro. Essas atualizações, conhecidas como *Link State Advertisements (LSAs)*, ocorrem quando o estado dos enlaces do roteador mudam. Quando o estado de um determinado enlace torna-se inacessível (o estado muda), o roteador manda uma atualização através do ambiente alertando todos os roteadores com os quais ele está diretamente conectado.

2.4 Do inglês: "Link-State Algorithm"

Em resumo, algoritmos de estado de enlace enviam pequenas atualizações para todos os vértices, enquanto algoritmos por vetor de distância enviam atualizações maiores somente para seus vizinhos.

Capítulo 3

Roteamento Baseado em Agentes

3.1 Resolução de Problemas Coletivamente

Agentes biológicos possuem muitas qualidades que são desejáveis em sistemas artificiais, de um ponto de vista de engenharia. Individualmente eles operam com regras simples, e coletivamente eles são capazes de formar padrões complexos como “soluções” para o problema de encontrar fontes de comida. Ainda mais, as formigas realizam isso grande parte do tempo sem um controle centralizado, o que significa que uma colônia de formigas é de diversas maneiras um sistema distribuído de computação descentralizada.

As regras simples seguidas pelas formigas e podem ser simplificadas para: [1]

- Formigas depositam feromônios a uma taxa aproximadamente constante enquanto viajam.
- Formigas são capazes de detectar diferenças na concentração de feromônios nas suas proximidades, e tendem a se mover na direção em que a concentração é maior.

Em adição, assim como em muitos sistemas naturais, existe certo grau de flutuação aleatória. Isto pode causar que formigas individuais sigam um caminho com pouca ou sem nenhuma concentração de feromônio. Ao que parece, essas simples regras em conjunto com as propriedades do feromônio, são suficientes para fazer surgir padrões organizados que são normalmente observados.

Isso pode ser demonstrado com um simples experimento [17, 18, 19], ilustrado na Figura

3.1. Ele serve para demonstrar como essas simples regras realizadas por cada indivíduo fazem surgir um processo coletivo de decisão. Nesse experimento, inicialmente não existe feromônio no caminho A nem no caminho B, então as formigas escolhem aleatoriamente algum dos caminhos a seguir. Nessa situação, uma formiga escolhe o caminho A e outra formiga escolhe o caminho B. No caminho para a comida, as formigas depositam uma trilha de feromônio que se difunde lentamente. Como o caminho B é mais curto que A, a formiga que escolheu B alcança a comida em um tempo menor que a formiga que escolheu o caminho A. Quando uma formiga encontra comida, ela retorna para o formigueiro com esta informação. A formiga no caminho B utiliza o mesmo caminho que realizou para retornar para o formigueiro. Obviamente, a formiga no caminho B deve retornar para o formigueiro antes que a formiga no caminho A. Nesse momento, uma terceira formiga deixa o formigueiro atrás da comida. Ela escolhe o caminho com base na intensidade de feromônio. A rota B é escolhida por ter uma trilha de feromônio mais forte. O resultado é que após um certo período de tempo, mais formigas utilizam o caminho mais curto. Isso significa que após um certo período, a concentração de feromônio do caminho mais curto é muito maior que a do caminho mais longo. Isso leva a um processo de realimentação positiva, onde após algum tempo, a maioria das formigas utiliza o caminho mais curto. Dessa forma, as formigas coletivamente fizeram uma “decisão”, e, mais importante, a decisão foi claramente a ótima.

Na prática, o ambiente na qual se encontra o caminho se modifica em tempo real. Por exemplo, caso o caminho B seja interrompido por algum obstáculo ou outra razão não esperada, Figura 3.2, o feromônio irá evaporar com o tempo e algumas formigas acabam escolhendo alguns outros caminhos aleatoriamente. Essas formigas são responsáveis pela exploração de novos caminhos. Dessa forma, quando o caminho B é interrompido, as formigas têm uma chance de encontrar o caminho C para a comida.

3.2 Desenvolvimento de Agentes Artificiais

Esse tipo de comportamento para resolver o problema coletivamente, descrito na seção 3.1, inspirou a tentativa de transladar os aspectos essenciais dos agentes biológicos para agentes artificiais na resolução de problemas computacionais, levando ao desenvolvimento da inteligência de enxames e da meta-heurística da otimização por colônia de formigas.

Figura 3.1 - Demonstração da capacidade das formigas de encontrar um caminho ótimo do formigueiro até a fonte de comida. (a) inicialmente não existe nenhum feromônio no caminho e as formigas escolhem aleatoriamente o caminho a seguir. (b) como o caminho B é mais curto, essa formiga retorna primeiro para o formigueiro. (c) a terceira formiga escolhe o caminho com base na intensidade de feromônio, escolhendo a rota B. (d) a maioria das formigas acaba utilizando o caminho mais curto. Retirado de [24].

Figura 3.2 - Caso o caminho B seja interrompido, as formigas conseguem explorar o ambiente para encontrar outros caminhos de menor distância até a fonte de comida. Retirado de [24].

3.2.1 Inteligência de Enxames

Inteligência de Enxames^{3.1} é uma propriedade de sistemas onde o comportamento coletivo de agentes (não-sofisticados) interagindo localmente com o seu ambiente provoca o surgimento de padrões globais coerentes e funcionais. Ela possibilita a base com a qual é possível resolver problemas coletivamente (distribuídos) sem a necessidade de um controle central ou modelo global [17].

Essa nova área de pesquisa científica emergiu do estudo do comportamento de espécies como formigas, abelhas, vespas, peixes e pássaros, onde cada indivíduo da colônia possui suas próprias tarefas, e ainda o grupo como um todo aparece altamente organizado, sem uma necessidade aparente de qualquer tipo de supervisão das atividades de integração dos indivíduos. Todas essas espécies possuem indivíduos com um comportamento social relativamente simples, mas que em conjunto podem atingir um comportamento coletivo complexo por interação. Isso permite resolver problemas com objetivos globais de uma maneira mais eficiente do que se um único indivíduo tentasse resolver. É fácil perceber que o comportamento do enxame é rigorosamente associado ao comportamento dos indivíduos, ou seja, o comportamento coletivo dos indivíduos determina o comportamento do enxame. Por outro lado, o comportamento do enxame determina as condições sobre as quais cada indivíduo realiza as suas ações. A interação social entre indivíduos é essencial para o comportamento coletivo.

3.1 Do inglês: Swarm Intelligence. Também poderia ser traduzido como Inteligência do Formigueiro.

3.2.2 Otimização por Colônia de Formigas

A otimização por colônia de formigas é uma área de pesquisa dentro da inteligência de enxames, sendo provavelmente a mais popular e a que obteve mais sucesso. A metaheurística da otimização por colônia de formigas é uma estrutura de multi-agentes para otimização combinatorial cujos principais componentes são [3]: um conjunto de *agentes tipo formigas*, o uso de *memória* e *decisões estocásticas*, e estratégias *coletivas* e de *aprendizado distribuído*.

A peculiaridade das colônias de formigas é que elas consistem de um grande número de indivíduos nas quais diferentes tipos morfológicos existem que são especializados para determinadas tarefas. Coletivamente, as formigas desempenham tarefas complexas como construir estruturas ótimas para o ninho, proteger a rainha e as larvas, limpar o ninho, achar as melhores fontes de comida e otimizar as estratégias de ataques. Todas estas atividades acontecem sem a existência de um comando central.

3.2.3 Conceito de Estigmergia

A interação social entre indivíduos pode ser direta ou indireta. A interação direta é o contato direto visual, auditivo ou químico entre os indivíduos. A interação social indireta ocorre quando um indivíduo modifica o ambiente e o comportamento futuro dos outros indivíduos é influenciado pela mudança do ambiente. Esse tipo de interação é chamado de stigmergia^{3.2}. Desta forma, as atividades ocorrem distribuídamente, sem a necessidade de um comando central.

Dois diferentes tipos de estímulos para a stigmergia podem ser distinguidos. As características físicas do ambiente podem ser modificadas, como a escavação de um buraco no local. Ao fazer isso, as outras formigas podem aumentar o buraco. Esse tipo de estímulo é chamado sematectônico^{3.3}. No segundo tipo: as formigas modificam os estímulos locais e, portanto, o comportamento de outras formigas. Por exemplo: ao depositar feromônios em certo local. Isto é chamado de stigmergia baseada em sinal. Como exemplo para a stigmergia baseada em sinal tem-se o problema de se encontrar o melhor caminho entre o formigueiro e uma fonte de comida. Nesse caso, as formigas tendem a seguir o caminho com a maior trilha de feromônios.

3.2 Do inglês: “stigmergy”. Esse termo é um pouco mais confuso para tradução, uma vez que não se pode encontrar uma tradução, ou um equivalente em português. Nesse caso, aplicou-se um neologismo a partir do termo em inglês para se obter o termo em português.

3.3 Do inglês: “sematectonic.”.

Assim, em cada intersecção elas verificam qual o caminho que possui a trilha mais forte e com uma alta probabilidade seguem esse caminho. Entretanto, algumas formigas não seguem esse caminho, elas seguem caminhos com menos feromônios ou até sem feromônios. Esse comportamento permite que as formigas possam investigar outras alternativas para encontrar melhores caminhos.

A persistência da trilha de feromônios que elas liberam depende da taxa com a qual as formigas liberam o feromônio, da quantidade depositada e da evaporação de feromônio no ambiente. Assim, se diversas formigas utilizarem o mesmo caminho haverá uma maior concentração de feromônios nesse caminho.

3.3 Isomorfismo entre Agentes Biológicos e Agentes Artificiais

A aplicação da meta-heurística a redes de comunicação é relativamente simples. A tabela 3.1 lista as analogias entre agentes reais e artificiais na aplicação de roteamento de redes de comunicação. Dessa forma, algoritmos que tomam como inspiração o comportamento das formigas para encontrar o caminho mais curto definem um conjunto de formigas artificiais que se movem sobre um grafo que representa a instância do problema. Enquanto se movem elas constroem soluções e modificam a representação do problema pela adição de informações coletadas.

Tabela 3.1 - *Analogias entre agentes biológicos e agentes artificiais [1].*

Agentes biológicos	Agentes artificiais
Formigas	Pacotes com informação
Trilhas	Enlaces da rede
Intersecções da trilha	Nós da rede
Concentração de feromônio	Pesos probabilísticos
Deposição de feromônio	Incremento no peso
Evaporação de feromônio	Decremento do peso

Assim, o feromônio natural de estigmergia é substituído por uma estigmergia artificial que pode ser modelada por computadores. A estigmergia artificial pode ser definida como a comunicação indireta que ocorre por modificações numéricas no estado do ambiente que é somente

acessível localmente pelos agentes de comunicação. Para os algoritmos de roteamento, a estigmergia artificial é armazenada nas tabelas de roteamento.

3.3.1 Agentes Inteligentes

Apesar de no AntNet, que será estudado adiante, os agentes se comportarem como pacotes, existe uma importante diferença conceitual entre os agente e os pacotes de dados: os agentes simulam pacotes com o objetivo de explorar a rede de uma maneira controlada (descobrimo e testando seus caminhos). Agentes não pertencem a aplicações de usuários, e são portanto livres para explorar a rede. A abordagem clássica é baseada normalmente na observação passiva de tráfego da rede: os nós observam o fluxo local dos dados, constroem estimativas locais dessas observações e propagam essas estimativas para outros nós. Os algoritmos baseados em agentes complementam a observação passiva do tráfego local com uma exploração ativa baseada nas repetidas simulações de Monte Carlo realizadas pelos agentes. Dessa maneira, as tabelas de roteamento são construídas e mantidas com base na observação do comportamento dos pacotes de dados e dos agentes gerados pelo próprio controle do sistema.

3.3.2 Tabelas de Roteamento e Tabelas de Feromônio

Uma tabela de roteamento é uma tabela convencional com linhas e colunas que pertence a um vértice da rede. Cada vértice da rede possui uma tabela de roteamento local armazenada. Para cada vértice vizinho que pode ser alcançado de um vértice a tabela possui uma coluna. E para cada vértice da rede exceto ele mesmo a tabela possui uma linha. As entradas na tabela são numéricas com valores entre 0 e 1, que correspondem à probabilidade de se alcançar outros vértices representados pelas linhas através de um determinado vértice vizinho.

As probabilidades na tabela de roteamento podem ser comparadas com a concentração da trilha de feromônios. Quanto mais alta a probabilidade, mais forte a trilha de feromônios. Perceba que as tabelas de roteamento possuem somente informações locais sobre as melhores rotas e nenhuma informação global.

Apesar de normalmente se utilizar uma única tabela de roteamento tanto para os pacotes de dados quanto para os agentes, uma outra possibilidade é separar a política de roteamento de dados da política de roteamento dos agentes. Dessa forma, os agentes utilizam sua própria tabela de roteamento (tabela de feromônio), enquanto os pacotes são roteados utilizando a tabela

de roteamento para dados, que é derivada da tabela de feromônio de tal forma que o fluxo de dados utilize somente os melhores caminhos descobertos até o momento. Dessa forma a política de exploração dos caminhos fica convenientemente separada da política de utilização dos caminhos.

3.3.3 *Calculo das Probabilidades*

Ainda é necessário determinar como as probabilidades nas tabelas de roteamento são calculadas. Na natureza as formigas depositam os feromônios e assim produzem trilhas entre o formigueiro e a fonte de comida. Em um computador o feromônio é substituído por uma estigmergia artificial, as probabilidades em uma tabela de roteamento. Para se calcular e atualizar essas probabilidades, agentes inteligentes são introduzidos para substituir as formigas. A maioria dos algoritmos inspirados em formigas se diferenciam pelo número de diferentes agentes e pelas diferentes funções realizadas por esses agentes ao passar pelos vértices. Eles se movem através da rede de vértice em vértice e através dos enlaces existentes, comunicando-se entre si de uma forma indireta através da leitura e escrita das tabelas de roteamento.

De uma perspectiva de engenharia, é desejável também aumentar os agentes artificiais com características adicionais não possuídas pelos agentes biológicos, para se evitar certas deficiências dos sistemas naturais.

Como exemplo [1], considerando o experimento da ponte dupla, descrito na seção 3.1, a realimentação positiva que serve para favorecer a escolha do caminho mais curto é essencialmente um efeito transitório, derivado do fato de que a concentração de feromônio aumenta a uma taxa maior no caminho mais curto. Uma vez estabelecido esse caminho, caso um caminho mais curto torne-se disponível, nem sempre as formigas conseguem trocar, e continuam a viajar no mesmo caminho. Esse efeito de estagnação pode ser evitado caso os agentes tenham a habilidade de depositar o feromônio em quantidades que refletem o comprimento do caminho sobre o qual estão viajando.

Este comportamento está claramente além da capacidade das formigas naturais, mas pode ser facilmente introduzido em agentes artificiais, onde, ao invés de depositar o feromônio artificial a uma taxa constante, os agentes artificiais depositam o feromônio em quantidades que são inversamente proporcionais ao comprimento associado ao caminho utilizado. Isso é realizado programando os agentes para refazerem os seus passos e depositarem o feromônio somente

após um caminho ter sido completado, e a métrica do caminho ser conhecida.

Apesar do fato de a arquitetura dos algoritmos ser relativamente simples, é necessário que o projeto de cada componente do algoritmo (avaliação dos caminhos, utilização das informações heurísticas, atualização do feromônio, etc.) sejam cuidadosamente realizados para se obter um algoritmo que não seja somente uma prova de conceito, mas um algoritmo capaz de fornecer performances comparáveis ou melhores que os outros algoritmos em condições realistas para um grande conjunto de cenários.

3.4 Algoritmos Propostos na Literatura

Diversos algoritmos que utilizam algum tipo de agente artificial foram propostos na literatura. Eles se diferenciam por grau de sofisticação, detalhes da implementação e objetivos. Entretanto, todos eles têm em comum a idéia de utilizar agentes móveis que atravessam a rede, e adaptativamente modificam as tabelas de roteamento.

Tabela 3.2 - Algoritmos propostos na literatura para o problema de roteamento em redes de comunicação utilizando agentes, tanto para o roteamento em redes orientadas a conexão quanto para redes não-orientadas a conexão.

Algoritmo	Autores	Ano	Ref.
ABC	Shoonderwoerd et al.	1996	[4]
AntNet e AntNet-FA	Di Caro e Dorigo	1997	[3,5,6,7,8,9]
AntNet-FS	Di Caro e Dorigo	1997	[3]
Regular Ants	Submanian et al.	1997	[18]
ASGA	White et al.	1998	[18]
ABC-smart ants	Bonabeau et al.	1998	[18]
CAF	Snyers e Kuntz	1998	[18]
ABC-backward	van der Put e Rothkrantz	1998	[18]
AntHocNet	Di Caro et al.	2004	[10,11]
BeeHive	Farooq et al.	2004	[23,22]

Capítulo 4

AntNet

O AntNet é um algoritmo baseado na metáfora da otimização por colônia de formigas originalmente proposto para o roteamento distribuído multicaminho e adaptativo ao tráfego em redes [3,5,6,7,8,9]. O seu projeto foi baseado nas idéias gerais sobre otimização baseada em formigas, como no trabalho de Schoonderwoerd et al. [4], que foi a primeira aplicação de algoritmos inspirados pelo comportamento de formigas para a tarefa de roteamento (em redes telefônicas). O AntNet atualmente representa uma família de algoritmos cujo comportamento é baseado na utilização de agentes móveis, que realizam uma simulação Monte Carlo conduzida por feromônios e atualizam os caminhos conectando os nós destinos com os nós fontes.

O mais antigo dos algoritmos é o AntNet, proposto em [7,5,9], sendo posteriormente atualizado e melhorado pelo AntNet-FA [8]. As últimas versões do AntNet são o AntHocNet [10,11,3], que é um algoritmo para roteamento em redes móveis ad hoc e o AntNet+SELA [3], um algoritmo para roteamento QoS em redes ATM.

4.1 Algoritmo AntNet

O AntNet é composto por dois conjuntos de *agentes móveis homogêneos*, chamados de *formigas em avanço* e *formigas em retorno*, ou *agentes em avanço* e *agentes em retorno*. Agentes de cada um dos conjuntos possuem a mesma estrutura básica, mas diferem em como são situados no ambiente, ou seja, eles podem sentir diferentes entradas e podem produzir diferentes saídas. Dessa forma, o algoritmo do AntNet pode ser informalmente descrito por [5,6,7,8,9]:

- Em intervalos regulares, cada vértice da rede s lança um agente móvel, um *agente em avanço*, $F_{s \rightarrow d}$ com um vértice de destino d aleatório. Os destinos são escolhidos para igualar os padrões de tráfego atuais.
- Cada agente seleciona o vértice para o próximo salto utilizando a informação armazenada na tabela de roteamento. O caminho é selecionado seguindo um esquema aleatório, proporcional à probabilidade de cada um dos vértices vizinhos, ou, com uma pequena probabilidade, utilizando a mesma probabilidade de seleção para cada um dos vértices vizinhos. No caso proporcional, caso o vértice escolhido já tenha sido visitado, uma seleção aleatória uniforme entre os vizinhos é aplicada.
- O identificador de cada vértice visitado k e o tempo decorrido desde o seu tempo de lançamento até chegar ao k -ésimo vértice é colocado em uma pilha de memória $S_{s \rightarrow d}(k)$ carregada pelo agente.
- Se um ciclo for detectado, isto é, se o agente for forçado a retornar para um vértice já visitado, os vértices do ciclo são retirados da memória da pilha do agente, e toda a memória sobre eles é destruída.
- Quando o vértice destino d é alcançado, o agente $F_{s \rightarrow d}$ gera um outro agente (*agente em retorno*) $B_{d \rightarrow s}$, transferindo para ele toda a sua memória.
- O agente em retorno utiliza o mesmo caminho que o correspondente agente em avanço, só que na direção contrária. Em cada vértice k através do caminho ele desempilha sua pilha $S_{s \rightarrow d}(k)$ para saber qual o vértice do próximo salto.
- Chegando em um vértice k vindo de um vértice vizinho f , o agente em retorno atualiza as estruturas de dados mantidas em cada vértice: a tabela de roteamento prt_k e um modelo paramétrico m_k que armazena as informações estatísticas do caminho.

A estrutura geral do algoritmo é relativamente simples [3]: durante a fase de avanço, cada agente móvel constrói um caminho a partir de uma sequência de escolhas baseadas em uma política estocástica parametrizada por informações locais do feromônio e heurística (o comprimento da fila local do enlace). Uma vez chegado ao destino, começa a fase de retorno. O agente refaz o caminho e em cada nó ele avalia o caminho seguido e atualiza a informação de roteamento local.

4.1.1 Estruturas de Dados

Dessa forma, o AntNet possui três estruturas de dados em cada roteador k :

Figura 4.1 - Estruturas de dados utilizadas pelos agentes no AntNet para o caso do nó com L vizinhos e a rede com N nós. A tabela de roteamento (feromônio) é organizada como em algoritmos de vetor de distância, mas as entradas não são distâncias mas probabilidades indicando a “qualidade” relativa dos enlaces possíveis para o próximo salto. A tabela de roteamento de dados é obtida a partir da transformação da tabela de feromônio. O modelo paramétrico armazena as informações relativas da distribuição do tráfego, para o cálculo da qualidade relativa dos diferentes caminhos. O estado atual das filas dos enlaces também é utilizado pelo AntNet. Retirado de [24]

Matriz de Roteamento (feromônio) p_{rt_k} organizada como em algoritmos de vetor de distância, mas cujos valores representam a quantidade de feromônio depositada pelos agentes. Consiste de uma tabela cujos valores $p_{rt_k}[i, d]$ expressam a “qualidade” de se escolher i como o próximo vértice quando o vértice destino é d . Têm-se a seguinte restrição:

$$\sum_{i \in \mathcal{N}_k} p_{rt_k}[i, d] = 1 \quad (4.1)$$

onde $d \in [1, N]$ e $\mathcal{N}$ é o número de vértices da rede, e $\mathcal{N}_k = \{vizinhos(k)\}$. Percebe-se facilmente que p_{rt_k} pode ser representada por uma tabela com $\mathcal{N} - 1$ linhas e $\mathcal{N}_k$ colunas. Dessa forma, os valores da tabela de feromônio podem ser interpretados como

a probabilidade de se selecionar um determinado enlace de saída para um determinado destino de acordo com o que foi aprendido até o momento pelos agentes.

Enlaces de Saída $\mathcal{L}_k$ Os enlaces de saída são estruturas independentes do AntNet, visto que fazem parte do roteador. O algoritmo AntNet simplesmente observa passivamente a dinâmica dos pacotes de dados. O AntNet utiliza a informação do estado das filas do roteador como um demonstrativo do que está acontecendo localmente no roteador no momento exato em que a decisão de roteamento está sendo tomada.

Modelo paramétrico estatístico $m_k(\mu_d, \sigma_d^2, W_d)$ A lista $m_k(\mu_d, \sigma_d^2, W_d)$ é uma estrutura de dados definindo um modelo estatístico paramétrico simples dos valores da distribuição de tráfego da rede como visto pelo vértice local k , onde μ_d é uma estimativa para a média do tempo de viagem para o vértice d e σ_d^2 é uma estimativa para a variância dessa medida. Dessa forma, o modelo é adaptativo e descrito por médias e variâncias calculadas sobre os tempos de viagem experimentados pelos agentes móveis. Além desses dois valores, existe também o valor de W_{best_d} que representa uma estimativa do tempo mínimo de viagem para o destino d do nó corrente, calculado em uma janela de observação móvel W_d . Dessa forma w_d representa o número máximo de amostras consideradas antes de se “ressetar” o valor de W_{best_d} .

prt_k e m_k podem ser vistos como uma memória local de longo prazo dos nós capturando diferentes aspectos da dinâmica da rede. O modelo m_k mantém estimativas absolutas de tempo/distância para todos os outros vértices, enquanto a tabela de roteamento fornece medidas probabilísticas relativas para cada enlace-destino. Pelo outro lado, o estado das filas dos enlaces $\mathcal{L}_k$ representam uma memória de curto prazo do tempo que se espera para atingir o nó vizinho.

Originalmente [5,6,7,8,9], o AntNet possuía essas três estruturas de dados em cada roteador k que eram utilizadas pelos agentes para realizar a comunicação indireta a partir do paradigma de estigmergia: uma tabela de roteamento prt_k , um modelo paramétrico estatístico m_k e as filas dos enlaces de saída $\mathcal{L}_k$. Na sua última descrição [3], a tabela de roteamento foi “dividida” em duas outras tabelas, como descrito na seção 3.3.2: a matriz de feromônio $\mathcal{T}_k$ (para o roteamento dos agentes) e a matriz de roteamento de dados $\mathcal{R}_k$. Isso ocorreu para um melhor projeto, visando separar a política de roteamento dos agentes da política de roteamento dos dados. Apesar dessa mudança, o algoritmo ainda possui o mesmo funcionamento, visto que nas versões anteriores os dados eram roteados a partir de uma transformação da tabela de

roteamento (feromônio), que é a mesma transformação utilizada para se calcular a tabela de roteamento dos dados. Percebe-se então que a mudança foi muito mais conceitual do que prática, mas importante, uma vez que pode abrir novas idéias para o futuro. Dessa maneira, neste trabalho, decidiu-se seguir a versão anterior da descrição do algoritmo, visando manter uma melhor semelhança com a descrição dos outros algoritmos apresentados, sem que com isso haja qualquer tipo de perda.

4.1.2 Descrição dos Agentes

Como descrito anteriormente, o AntNet é composto por dois tipos de agentes homogêneos móveis, chamados *formigas em avanço* e *formigas em retorno* ou *agentes em avanço* e *agentes em retorno*. Cada agente possui a mesma estrutura, mas eles são diferentemente posicionados no ambiente, isto é, eles podem perceber diferentes entradas e podem produzir diferentes saídas, independentemente. Os *agentes em avanço* são responsáveis por medir os tempos de latência e congestionamento dos enlaces sendo verificados. Os *agentes em retorno* são responsáveis por distribuir as informações acumuladas para os vértices percorridos no caminho de volta ao vértice fonte. Dessa forma, existe um efeito de *sobreposição* da ação dos agentes que possuem destinos diferentes, mas que percorrem, em algum momento, o mesmo vértice pelo caminho. A tabela 4.1 mostra a estrutura utilizada pelos agentes.

Tabela 4.1 - Estrutura dos Agentes do AntNet. Tanto os agentes em avanço quanto os agentes em retorno utilizam a mesma estrutura básica. Eles simplesmente percebem diferentes entradas e saídas, tendo o seu comportamento diferenciado na rede por essas diferenças.

Propriedade	Descrição
<i>srcAddr</i>	endereço da fonte do agente
<i>destAddr</i>	endereço do destino do agente
<i>type</i>	tipo de agente
<i>hops</i>	contagem de saltos desde a fonte
<i>vNodes</i>	uma matriz com hops + 1 entradas, uma para cada salto desde a fonte.
<i>vNodes[i].address</i>	endereço do vértice i
<i>vNodes[i].timestamp</i>	somente para agentes de duas vias: registro de tempo na direção de vinda.

4.1.2.1 Criação dos Agentes

Os agentes em avanço $F_{s \rightarrow d}$ são criados e lançados de cada um dos vértices s da rede em intervalos regulares Δt com um destino d , para descobrir caminhos possíveis e de baixo custo e investigar o estado do carregamento atual da rede. Os agentes em avanço dividem a mesma fila que os pacotes de dados, assim eles experimentam as mesmas condições de tráfego. Eles representam uma simulação dos pacotes de dados, sendo um experimento com o objetivo de coletar informações úteis.

4.1.2.2 Seleção dos Destinos

Os destinos são localmente selecionados, de acordo com os padrões de tráfego de dados gerados: se $f_{s \rightarrow d}$ é uma medida (em bits ou número de pacotes) do fluxo de dados de $s \rightarrow d$, então a probabilidade de criar no vértice s um agente em avanço com o vértice d como destino é

$$p_d = \frac{f_{s \rightarrow d}}{\sum_{d'=1}^{\mathcal{N}} f_{s \rightarrow d'}}. \quad (4.2)$$

Dessa forma, os agentes adaptam a sua atividade de exploração com a variação da distribuição do tráfego de dados.

Ao viajar através dos vértices para o seu destino, os agentes mantêm na memória o caminho percorrido e as condições de tráfego encontradas. O identificador de cada vértice k e o tempo decorrido desde o lançamento até a chegada ao k -ésimo vértice são armazenados na pilha de memória $S_{s \rightarrow d}(k)$.

A taxa de geração de agentes $\frac{1}{\Delta t}$ determina o número de experimentos realizados. É necessário manter o número de experimentos realizados alto para diminuir as estimativas das variâncias, mas também não se pode manter esse valor tão alto que ele tenha um impacto sobre a rede.

4.1.3 Roteamento dos Agentes em Avanço

Os agentes em avanço dividem a mesma fila de espera no roteador que os pacotes de dados. Dessa maneira, ao cruzar uma área congestionada, eles também são atrasados. Isso possui um duplo efeito: o tempo de viagem vai aumentar e os incrementos de probabilidade no retorno serão pequenos e com um atraso maior.

Em cada vértice k , cada agente com destino d seleciona um vértice n para prosseguir entre os vizinhos que ele não visitou, ou sobre todos os vizinhos no caso de todos terem sido previamente visitados. O vizinho n é selecionado com uma probabilidade P_{nd} calculada como a soma normalizada das probabilidade nas entradas $prt_k[n, d]$ da tabela de roteamento com um fator de correção heurística l_n levando em conta o estado (comprimento) da fila do n -ésimo enlace do vértice atual k :

$$P_{nd} = \frac{prt_k[n, d] + \alpha l_n}{1 + \alpha(|\mathcal{N}_k| - 1)} \quad (4.3)$$

A correção heurística l_n é um valor normalizado no intervalo $[0, 1]$ proporcional ao comprimento q_n (em bits esperando para serem enviados) da fila do enlace conectando o vértice k com seu vizinho n :

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|\mathcal{N}_k|} q_{n'}} \quad (4.4)$$

O valor α pesa a importância da correção heurística com respeito às probabilidades armazenadas na tabela de roteamento, l_n reflete o estado instantâneo das filas no vértice, e assumindo que o processo de utilização das filas é quase estacionário ou variando lentamente, l_n fornece a quantidade medida associada com o tempo na fila de espera. Os valores da tabela de roteamento, por outro lado, são a saída de um processo de aprendizagem contínuo e capturam os valores correntes e passados do estado da rede como visto pelo vértice local. Corrigindo esses valores com o valor de l_n torna o sistema mais reativo, e ao mesmo tempo evita seguir todas as flutuações da rede. As decisões do agente são tomadas com base na combinação de processos de aprendizagem de longo termo e predições heurísticas instantâneas.

Caso um ciclo seja detectado, isto é, caso o agente seja forçado a retornar para um vértice já visitado, os vértices do ciclo são retirados da memória e destruídos. Se o ciclo tiver durado mais do que a metade do tempo de vida do agente antes de ter entrado no ciclo (isto é, o ciclo é maior que metade da idade do agente), o agente é destruído. De fato o agente perdeu um grande tempo por causa de uma sequência de decisões ruins, e seria contra-produtivo utilizá-lo para atualizar as tabelas de roteamento.

Assim com todos os vizinhos já visitados, o agente escolhe um próximo salto para executar através de uma seleção aleatória entre todos os vizinhos, sem nenhuma preferência, mas

excluindo-se o nó pelo qual chegou ao nó k , o nó v_{i-1} . Dessa forma, tem-se que P_{nd} pode ser calculada por:

$$P_{nd} = \begin{cases} \frac{1}{|\mathcal{N}_k| - 1} & \forall n \in \mathcal{N}_k \wedge n \neq v_{i-1} \\ 0 & \text{caso contrário.} \end{cases} \quad (4.5)$$

Uma outra proposta realizada pelo autor do AntNet e implementada em [14] define também uma outra condição para a seleção uniforme entre todos os vizinhos. Nesse caso foi definida uma probabilidade δ com a qual o agente, mesmo sem ter visitado todos os vizinhos, decide-se por realizar a seleção uniforme ao invés da seleção aleatória proporcional que realizaria normalmente. Dessa forma, existe uma maior exploração da rede por parte dos agentes.

4.1.4 Roteamento dos Agentes em Retorno

Ao contrário dos agentes em avanço, os agentes em retorno possuem prioridade sobre os pacotes de dados para acelerar a propagação da informação acumulada. Esse último aspecto é extremamente importante: formigas em avanço utilizam as mesmas filas que os pacotes de dados, enquanto as formigas de retorno não, elas possuem prioridade sobre os pacotes de dados para aumentar a velocidade de propagação da informação acumulada.

Os agentes em retorno tomam o mesmo caminho utilizado pelo agente em avanço, mas na direção contrária^{4.1}. Em cada vértice k através do caminho eles desempilham sua pilha $S_{s \rightarrow d}(k)$ para verificar o próximo salto.

4.1.5 Roteamento dos Pacotes de Dados

Para realizar o roteamento dos pacotes de dados é utilizada a tabela de roteamento de dados $\mathcal{R}_k$. Essa tabela é calculada a partir da tabela de roteamento (feromônio) prt_k , atualizada pelos agentes, a partir da transformação por uma função exponencial $f(p) = p^\epsilon$, onde $\epsilon > 1$, que enfatiza valores com alta probabilidade e diminui os de baixa probabilidade. Esse mecanismo permite uma eficiente distribuição dos pacotes sobre todos os bons caminhos. A escolha de enlaces com uma probabilidade muito baixa é evitada por esse remapeamento dos valores de probabilidade. Por causa da simplicidade da operação, não se faz, normalmente, necessário

4.1 essa suposição necessita que todos os enlaces na rede sejam bidirecionais. Nas redes modernas essa é uma suposição aceitável.

manter esse espaço em memória, mas geralmente se calcula esse valor no momento em que ele se faz necessário.

4.1.6 Atualização das Estruturas de Dados

Ao chegar ao vértice k vindo de um vértice vizinho f , os agentes em retorno atualizam as duas principais estruturas de dados do vértice, o modelo local do tráfego m_k e a tabela de roteamento prt_k , para todas as entradas correspondendo ao vértice destino d (do agente em avanço). Com algumas precauções, a atualização é realizada também nas entradas correspondentes a cada vértice $k' \in S_{k \rightarrow d}$, $k' \neq d$ nos subcaminhos seguidos pelo agente $F_{s \rightarrow d}$ após visitar o vértice atual k . De fato, caso o tempo de viagem do sub-caminho seja estatisticamente “bom” (isto é, menor que $\mu + I(\mu, \sigma)$, onde I é uma estimativa do intervalo de confiança para μ), então o valor do tempo é utilizado para atualizar as correspondentes estatísticas e a tabela de roteamento. Caso contrário, os tempos de viagem dos sub-caminhos não são considerados satisfatórios, pelo mesmo sentido estatístico definido anteriormente, e não são utilizados por que não fornecem uma idéia correta do tempo de viagem para o vértice no sub-destino. De fato, todas as decisões de roteamento dos agentes em avanço são feitas somente em função do vértice destino. Dessa perspectiva, os sub-caminhos são efeitos secundários, e eles são intrinsecamente sub-ótimos por causa das variações locais no carregamento do tráfego.

Dessa forma, pode-se detalhar o procedimento seguido para atualizar as estruturas de dados: cada roteador k reitera as informações contidas sobre os vértices visitados na matriz $vNodes$, ou seja, na memória de cada um dos agentes em retorno. O roteador atualiza as estruturas de dados somente em dois casos, para se evitar atualizar a tabela de roteamento com sub-caminhos que apresentem uma latência maior do que algum caminho já encontrado:

1. Se a entrada na matriz $vNodes$ possui o mesmo endereço que o endereço de destino do agente de retorno;
2. Caso o tempo de latência $t_{i \rightarrow k}$ medido pelo agente do vértice i até o vértice k obedeça à

$$t_{k \rightarrow i} = t_i - t_k < I_{sup} \quad (4.6)$$

onde I_{sup} é uma estimativa conveniente para o limite superior do tempo de viagem dos agentes, sendo melhor definido na seção 4.1.6.3.

4.1.6.1 Atualização do Modelo de Tráfego Local

O modelo de tráfego local m_k é atualizado com os valores na pilha de memória $S_{s \rightarrow d}(k)$. O tempo decorrido para chegar ao vértice destino d (para os agentes em avanço), começando do vértice atual, é utilizado para atualizar as estimativas da média μ_d e variância σ_d^2 e o melhor valor sobre a janela de observação W_d . Dessa forma, um modelo paramétrico do tempo de viagem para o destino d é mantido.

Para cada destino d na rede, uma média e variância estimadas, μ_d e σ_d^2 , fornecem uma representação do tempo esperado de viagem e sua estabilidade. É utilizada uma estratégia exponencial para se calcular as estatísticas.

$$\begin{aligned}\mu_d &\leftarrow \mu_d + \eta(t_{k \rightarrow d} - \mu_d) \\ \sigma_d^2 &\leftarrow \sigma_d^2 + \eta((t_{k \rightarrow d} - \mu_d)^2 - \sigma_d^2)\end{aligned}\tag{4.7}$$

onde $t_{k \rightarrow d}$ é o novo tempo de viagem observado pelo agente do vértice k para o destino d .

A janela móvel de observação W_d é utilizada para calcular o valor de W_{best_d} , ou seja, o melhor (menor) tempo de viagem dos agentes para o destino d observado nas últimas W_d amostras. Após cada nova amostra, W_d é incrementada modulus $|W|_{max}$, onde $|W|_{max}$ é o tamanho máximo permitido para a janela de observação. Caso o contador W_d seja igual a zero, ou o último tempo medido seja menor que o atual valor de W_{best_d} , o valor de W_{best_d} é atualizado. Dessa forma, o valor W_{best_d} representa uma memória de curto prazo expressando um limite inferior empírico para uma estimação do tempo de viagem para o vértice d do vértice atual. Os valores de η na equação 4.7 e de W_{max} podem ser ajustados de tal maneira que as amostras efetivas para as médias exponenciais estejam diretamente relacionadas com aquelas utilizadas para monitorar W_{best_d} . De acordo com a expressão em [3], tem-se que o valor de W_{max} é ajustado por:

$$W_{max} = \frac{5c}{\eta}.\tag{4.8}$$

4.1.6.2 Atualização da Tabela de Roteamento

A tabela de roteamento p_{rt_k} é modificada pelo incremento da probabilidade $p_{rt_k}[f, d]$ (isto é, a probabilidade de se escolher o vizinho f quando o destino é d), e decrementando, por normalização, as outras probabilidades $p_{rt_k}[n, d]$. A quantidade de variação nas probabilidades

depende da medida da qualidade que é associada com o tempo de viagem $t_{k \rightarrow d}$ experimentado pelo agente em avanço, e é dada abaixo. A atualização acontece utilizando o único sinal de realimentação medido, o tempo de viagem experimentado pela formiga em avanço. Este tempo fornece uma indicação clara sobre a qualidade do caminho seguido, visto que é correspondente ao seu *comprimento* do ponto de vista físico (número de saltos, capacidade de transmissão dos enlaces utilizados, velocidade de transmissão, et cetera) e do ponto de vista de congestionamento do tráfego.

A medida de tempo t não pode ser associada com uma medida de erro exata, dado que não se sabem os tempos de viagem “ótimos”, que dependem do estado de carregamento de toda a rede. Dessa forma, t só pode ser utilizado como um sinal de reforço. É definido um reforço $r \equiv r(t, m_k)$ que é uma função da qualidade dos tempos de viagem observados com base no modelo de tráfego local. r é uma medida sem dimensão, pertencente ao intervalo $(0, 1]$, utilizado pelo vértice atual k como um reforço positivo para o vértice f do qual o agente em retorno $B_{d \rightarrow s}$ se origina.

Assim, a probabilidade $p_{rt_k}[f, d]$ é incrementada pelo valor do reforço a partir de:

$$p_{rt}[f, d] \leftarrow p_{rt}[f, d] + r(1 - p_{rt}[f, d]) \quad (4.9)$$

Dessa maneira, a probabilidade $p_{rt_k}[f, d]$ é aumentada por um valor proporcional ao valor de reforço recebido e ao valor anterior da probabilidade do vértice. As probabilidades $p_{rt_k}[f, d]$ para o destino d dos outros vértices vizinhos n recebem um reforço negativo por normalização. Isto é, seus valores são reduzidos para que a soma de probabilidades continue 1:

$$p_{rt}[n, d] \leftarrow p_{rt}[n, d] - r \cdot p_{rt}[n, d], n \in N_k, n \neq f \quad (4.10)$$

É importante realçar que todo caminho descoberto recebe um reforço positivo na sua probabilidade de seleção, e o reforço é (em geral) uma função não-linear da qualidade do caminho, como estimada utilizando o tempo de viagem associado. Dessa maneira, não somente o valor de r é importante, mas também a taxa de chegada de agentes.

4.1.6.3 Cálculo do Reforço

O reforço r é uma quantidade crítica que é determinada considerando-se três principais aspectos:

- os caminhos precisam receber um incremento na sua seleção probabilística proporcional à qualidade;
- a qualidade é uma medida relativa, que depende das condições de tráfego, que pode ser estimada a partir do modelo m_k ;
- é importante não seguir todas as flutuações do tráfego.

No trabalho original de [7], foram testadas diversas abordagens alternativas para o cálculo do reforço r , considerando-se diversas combinações lineares, quadráticas e hiperbólicas dos valores de $t_{d \rightarrow s}$ e m_k . Limitando-se a descrição para a forma funcional que forneceu os melhores resultados, pode-se calcular r como:

$$r = \theta_1 \left(\frac{W_{best_d}}{t_{x \rightarrow d}} \right) + \theta_2 \left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (t_{x \rightarrow d} - I_{inf})} \right) \quad (4.11)$$

onde W_{best_d} é o melhor tempo experimentado pelos agentes até o destino d , sobre a última janela de observação W_d . O tamanho máximo da janela (o número máximo de amostras consideradas antes de se “ressetar” o valor de W_{best_d}) é determinado com base no coeficiente η da equação 4.8. Como dito, η pesa o número de amostras efetivamente fornecendo uma contribuição para o valor de μ estimado, definindo um tipo de janela móvel exponencial. I_{sup} e I_{inf} são estimativas convenientes para os limites de um intervalo de confiança aproximado para μ . I_{inf} é ajustado para W_{best_d} , enquanto I_{sup} pode ser calculado por:

$$I_{sup} = \mu + z \left(\frac{\sigma}{\sqrt{|W_{max}|}} \right), \quad (4.12)$$

onde z equivale à:

$$z = \frac{1}{\sqrt{(1 - \zeta)}} \quad (4.13)$$

e ζ fornece o nível de confiança selecionado. Existe um nível de arbitrariedade no cálculo do intervalo de confiança, visto que ele é definido de uma maneira assimétrica e μ e σ não são estimativas aritméticas. De qualquer maneira, o que é preciso é uma estimativa rápida do valor médio e da sua dispersão, ou seja, um procedimento que forneça um resultado com significado mas cujo cálculo não gaste excessivos ciclos de processamento.

O primeiro termo na equação 4.11 simplesmente calcula a razão entre a viagem atual e a melhor viagem observada sobre a atual janela de observação. Esse termo é corrigido por um segundo termo, que calcula quão distante o valor t está de I_{inf} em relação ao intervalo de confiança,

isto é, considerando-se a estabilidade nos tempos das últimas viagens. Os coeficientes θ_1 e θ_2 pesam a importância de cada um dos termos. O primeiro termo é o mais importante, enquanto o segundo é um fator de correção.

O valor de r obtido da equação 4.11 é finalmente transformado a partir de uma função de compressão $s(x)$:

$$s(x) = \left(1 + e^{\frac{a}{x|\mathcal{N}_k|}}\right)^{-1} \quad (4.14)$$

onde $x \in (0, 1]$ e $a \in \mathbb{R}^+$, e r pode ser calculado por:

$$r \leftarrow \frac{s(r)}{s(1)}. \quad (4.15)$$

A compressão dos valores de r permite ao sistema ser mais sensível às boas recompensas (altos valores de r) enquanto tem uma tendência de saturar as más recompensas (valores de r próximos de zero). A escala é comprimida para baixos valores e expandida na parte superior. Dessa maneira é colocada uma ênfase sobre os bons resultados, enquanto os maus resultados têm um papel menor.

O coeficiente $\frac{a}{|\mathcal{N}_k|}$ determina uma dependência paramétrica do valor de reforço comprimido em relação ao número de vizinhos $|\mathcal{N}_k|$ do vértice de reforço k : quanto maior o número de vizinhos, maior é o reforço.

4.1.7 Resumo dos Parâmetros

Dessa maneira, pode-se resumir na tabela 4.2 todos os parâmetros utilizados. Esses parâmetros representam os valores utilizados pelos autores do algoritmo, Di Caro et al. [3,5,6,7,8,9] para os quais o AntNet apresentou os melhores desempenhos, além de alguns outros valores apresentados no trabalho de Farooq [3], que teve sugestões dos autores do AntNet.

Em [5,3] é lembrado ainda que esses valores talvez possam ser um pouco melhorados, visto que não foram exaustivamente testados, principalmente pela verificação de que o AntNet é robusto o suficiente para suportar variações nesses parâmetros sem uma perda de eficiência.

Tabela 4.2 - Sumário dos parâmetros utilizados no AntNet.

Parâmetro	Valor	Descrição	Seção
Δt	0.225	Intervalo de criação dos agentes.	4.1.2.1
α $\alpha \in [0, 1]$	0.35	Peso da correção heurística.	4.1.3
δ $\delta \in [0, 1]$	0.1	Probabilidade de seleção uniforme.	4.1.3
ϵ $\epsilon \geq 1$	1.0	Coefficiente de compressão da função exponencial.	4.1.5
η $\eta \in (0, 1]$	0.775	Coefficiente da média exponencial.	4.1.6.1
c $c \in (0, 1]$	0.3	Coefficiente do tamanho da janela móvel.	4.1.6.1
θ_1	0.7	Peso para a razão entre o tempo de viagem atual e o melhor tempo para cálculo do reforço.	4.1.6.3
θ_2	0.3	Peso para o valor da distância do valor de t de I_{inf} para o cálculo do reforço.	4.1.6.3
ζ	0.6	Coefficiente do nível de confiança atribuído.	4.1.6.3
a $a \in \mathbb{R}^+$	2.0	Coefficiente da função de compressão.	4.1.6.3

4.2 Algoritmo AntNet-FA

No AntNet os agentes em avanço utilizam as mesmas filas utilizadas pelo pacotes de dados. Dessa maneira eles se comportam como pacotes de dados e experimentam os mesmos tempos de viagem que os pacotes de dados experimentariam. O problema com essa abordagem é que se existe um congestionamento através do caminho percorrido, a jornada dos agentes em avanço pode ser consideravelmente longa. Assim, o sinal de reforço local vai ser pequeno e o agente vai ser atrasado para aplicar o reforço ao seu caminho. Por outro lado, agentes que seguiram caminhos menos congestionados vão atualizar suas tabelas de roteamento pelo caminho que seguiram mais cedo e com uma maior intensidade, “corrigindo” dessa maneira as escolhas dos pacotes de dados subsequentes.

Um problema com essa abordagem pode ocorrer com o seguinte cenário: um agente está seguindo um bom caminho, mas existe um congestionamento temporário no vértice atual onde o agente se encontra. O agente é então forçado a esperar por um longo tempo. Quando ele deixa o vértice, o problema do congestionamento já está resolvido, mas infelizmente o agente foi atrasado e todo o caminho não irá receber a recompensa apropriada. Um outro cenário que

pode ocorrer corresponde a um caminho congestionado logo após a passagem do agente. Nesse caso, o agente percorre o caminho rapidamente, mas infelizmente as informações medidas pelo agente não correspondem mais à realidade. Ambos os problemas podem ocorrer com uma probabilidade que aumenta com o aumento do tamanho da rede.

Dessa forma, dois problemas podem ser facilmente identificados:

- esperar na mesma fila com baixa prioridade com os pacotes de dados pode determinar o fato de que uma visão obtida do estado do tráfego pode estar completamente atrasada quando as tabelas de roteamento são atualizadas;
- o tempo medido pelos agentes em avanço no seu caminho até o destino e retorno para a atualização da tabela de roteamento ao retornar pode sofrer do mesmo problema, ou seja, a visão obtida do estado do tráfego pode estar novamente atrasada no momento de atualização das tabelas de roteamento.

Uma maneira de se evitar esse problema é:

- os agentes em avanço devem utilizar as filas de alta prioridade nos roteadores, assim como os agentes em retorno;
- os agentes em retorno podem atualizar as tabelas de roteamento nos vértices visitados utilizando estimativas dos tempos de viagem dos agentes.

De acordo com essas modificações, os agentes em avanço não simulam mais os pacotes de dados de uma maneira mecanicista. O tempo de viagem deles não reflete realisticamente os tempos de viagem dos pacotes de dados. De forma a se obter um valor de $t_{k \rightarrow l}$ que realisticamente estime o tempo de viagem que um pacote seguindo esse caminho iria experimentar é necessário utilizar um modelo para a dinâmica da depleção das filas dos enlaces.

Di Caro et al. [8,3] desenvolveram um modelo simples, mas que forneceu os melhores resultados. A dinâmica da depleção das filas dos enlaces é modelada em termos de um processo uniforme e constante $\mathcal{D}$ que depende somente da largura de banda B_l do enlace. Isto é, tanto o remetente do processo quanto o consumidor do processo do outro lado do enlace não diminuem a depleção do enlace. De acordo com essas suposições, a transmissão do pacote na fila do enlace $\mathcal{L}_l^k$ para o vizinho l é completada após um intervalo de tempo definido por:

$$\mathcal{D}_l^k(q_l; B_l; d_l) = d_l + \frac{q_l}{B_l}, \quad (4.16)$$

onde q_l é o número total de bits contidos no pacote esperando na fila $\mathcal{L}_l^k$, enquanto B_l e d_l são respectivamente a largura de banda do enlace (bits/s) e atraso de propagação (s).

Figura 4.2 - Agentes em avanço no AntNet e agentes em avanço no AntNet-FA. Os agentes em avanço no AntNet-FA utilizam uma fila de alta prioridade nos roteadores atualizando as tabelas de roteamento com estimativas dos tempos de viagem.

De acordo com esse modelo de depleção, a estimativa do tempo $t_{k \rightarrow i}$ que um pacote de mesmo tamanho s_a que o agente em avanço demoraria para chegar ao vizinho l pelo qual o agente em retorno está vindo é calculado por:

$$t_{k \rightarrow i} = \mathcal{D}_l^k(q_l + s_a; B_l; d_l). \quad (4.17)$$

Tanto em AntNet quanto em AntNet-FA os agentes em avanço utilizam uma política estocástica para descobrir possíveis novos caminhos: no AntNet os agentes em avanço comportam-se exatamente como os pacotes de dados e os atrasos experimentados por eles são utilizados pelos agentes em retorno para determinar a qualidade do caminho percorrido, enquanto que no AntNet-FA, os agentes em avanço rapidamente descobrem novos caminhos que são analisados pelos agentes em retorno utilizando os tempos de viagem local estimado a partir do modelo $\mathcal{L}$. Perceba também que os agentes em avanço também são mais “leves” de um ponto de vista computacional, visto que eles não carregam mais na memória qualquer informação sobre os tempos de viagem $t_{s \rightarrow k}$.

Comparado com o modelo anterior, onde os agentes estavam “andando” lentamente sobre as “estradas” congestionadas, aqui os agentes, figurativamente, estão “voando” sobre as filas de dados, e rapidamente atingindo o objetivo. Dessa maneira, pode-se sumarizar as diferenças entre o AntNet e o AntNet-FA:

- os agentes em avanço possuem prioridade na fila, assim como os agentes em retorno;
- eles não carregam na memória qualquer informação sobre os tempos experimentados;

- os agentes em retorno atualizam as tabelas de roteamento nos vértices visitados estimando o tempo de viagem dos agentes.

Capítulo 5

Trail Blazer

O Trail Blazer (TB) é um algoritmo de roteamento proposto por Gabber e Smith [13] que tem como função diminuir o congestionamento das redes a partir de decisões locais baseadas em medidas de latência coletadas por pacotes exploradores. O TB diferencia-se do AntNet principalmente por:

- o TB foi projetado para ser uma extensão dos atuais protocolos de estado de enlace como o OSPF ou IS-IS, assumindo que sempre existe um protocolo de estado de enlace ativo. O protocolo de estado de enlace inicializa a tabela de probabilidades dos enlaces com as informações do caminho mais curto e ajusta essas após mudanças na topologia, como falhas nos enlaces ou adições;
- o TB utiliza operações mais simples que o AntNet, tendo, portanto, uma complexidade computacional menor, o que pode ser um fator decisivo em redes de grande porte, onde não seria viável o uso de algoritmos de complexidade maior;

Dessa forma, o TB melhora os protocolos de estado de enlace pela adaptação das decisões de roteamento ao escolher enlaces com a menor latência para o destino. Dessa maneira, o tráfego é direcionado através de caminhos menos congestionados, que não sofrem de longos atrasos em filas.

5.1 Algoritmo Trail Blazer

O TB pode operar de dois modos: de *mão única* ou *mão dupla*^{5.1}. O TB de mão única envia agentes a partir de destino de fluxos ativos em direção às suas respectivas fontes. Dessa maneira, os agentes acumulam a latência até o destino enquanto viajam em direção à fonte, e utilizam essa informação sobre a latência acumulada para atualizar as tabelas de roteamento nos roteadores intermediários.

No caso do TB de mão dupla, enviam-se agentes da fonte do fluxo ativo em direção aos seus respectivos destinos. Os agentes de duas vias mantêm um registro de tempo local^{5.2} de todos os vértices intermediários no caminho para o destino. Ao chegarem no destino, mudam de direção e refazem exatamente o mesmo caminho em direção à fonte. No caminho de volta, os agentes vão atualizando as tabelas de roteamento a partir do cálculo da latência em cada um dos roteadores, subtraindo o registro anterior armazenado do tempo de chegada.

A vantagem dos agentes de mão única é que todo agente atualiza as tabelas de roteamento dos vértices intermediários, mesmo que o pacote seja perdido por algum motivo. Isso não ocorre com os agentes de mão dupla, visto que eles precisam chegar ao seu destino, para então retornar e começar a atualizar as tabelas de roteamento. Entretanto, os agentes de mão única apresentam a desvantagem de serem roteados utilizando-se a probabilidade de enlace na direção reversa, isto é, do destino para a fonte. Estas tabelas podem ser diferentes das tabelas da fonte até o destino devido a tráfego assimétrico. Essa é exatamente a vantagem dos agentes de mão dupla, que sempre utilizam o caminho da fonte até o destino.

Existem dois tipos de agentes: *agentes de melhor rota*^{5.3} e *agentes exploradores*^{5.4}. Os agentes de melhor rota viajam através dos caminhos existentes com maior probabilidade, enquanto aos agentes exploradores é permitido se desviarem da melhor rota ocasionalmente. Os agentes exploradores preferem ainda os enlaces com pouco congestionamento aos enlaces congestionados. Um agente nunca seleciona um enlace de saída que já tenha percorrido (um loop). Se somente existirem esses enlaces disponíveis, o agente se auto-destrói.

Outro detalhe de implementação é que o TB gerencia falhas de enlaces pela atualização da

5.1 Do inglês: “one way” ou “two way”.

5.2 Do inglês: “local time stamp”.

5.3 Do inglês: “best-path scout agent”.

5.4 Do inglês: “exploratory scout agent”.

tabela de probabilidade de enlaces com a utilização da informação do protocolo de estado de enlace. Dessa maneira, o TB é rapidamente notificado em todos os vértices sobre mudanças na topologia, recompondo as tabelas de roteamento para refletir a nova topologia. Posteriormente, os agentes irão atualizar as tabelas para refletir as novas latências da rede.

5.1.1 Estruturas de Dados

O TB possui duas estruturas de dados em cada roteador: uma tabela de roteamento prt_k com as probabilidades calculadas e uma tabela avg_k com o atraso de transmissão médio.

Tabela de roteamento prt_k possui $\mathcal{N} - 1$ linhas, uma para cada destino. Cada linha possui $\mathcal{N}_k$ entradas, uma para cada enlace de saída do roteador. O elemento $prt_k(i, d)$ é a probabilidade de enviar um pacote com destino d no enlace de saída i . Portanto, como no AntNet, tem-se que:

$$\sum_{i \in \mathcal{N}_k} prt_k[i, d] = 1. \quad (5.1)$$

A **tabela** avg_k mantém a média do atraso de transmissão para os destinos a partir do vértice atual k . avg_k possui $\mathcal{N} - 1$ elementos, uma para cada destino. O elemento $avg_k(d)$ é o atraso de transmissão médio do vértice atual para o destino d , que é calculado com base nos M últimos agentes. Foi utilizado o valor de $M = 10$, que é o valor utilizado por Gabber; Smith [13], correspondendo a um compromisso entre memória consumida e sensibilidade às medidas recentes.

5.1.1.1 Inicialização da Tabela de Roteamento

O valor inicial de prt_k é calculado a partir da informação do caminho mais curto obtida pelo protocolo de estado de enlace. Seja $SP(d)$ o conjunto de enlaces de saída que pertencem a um ou mais dos caminhos mais curtos do nó atual para o destino d , calculado pelo algoritmo sobre o qual o TB opera. O valor de prt_k é:

$$prt_k[i, d] = \begin{cases} \frac{1}{|SP(d)|} & i \in SP(d) \\ 0 & \text{caso contrário.} \end{cases} \quad (5.2)$$

Sempre que o protocolo de estado de enlace descobrir uma mudança de topologia, como após uma falha de enlace ou adição de novos enlaces, o TB combina as informações descobertas

pelo algoritmo do caminho mais curto com os valores históricos acumulados em prt_k . O novo valor de prt_k é:

$$prt_k[i, d] = \begin{cases} \beta prt_k[i, d] + \frac{1 - \beta}{|SP(d)|} & i \in SP(d) \\ \beta prt_k[i, d] & \text{caso contrário.} \end{cases} \quad (5.3)$$

Percebe-se que o parâmetro β controla a quantidade de informação acumulada que é mantida após uma mudança de topologia, representando uma memória para o algoritmo.

5.1.2 Descrição de Agentes

O TB gera dois tipos de agentes: agentes de melhor rota e agentes exploradores. Eles são gerados a partir da contagem de um número específico N de pacotes, e diferenciam-se somente pela maneira em que são roteados. Agentes de melhor rota são roteados similarmente aos pacotes de dados de acordo com as probabilidades na tabela prt . Esses agentes nunca tentam novos caminhos, e são usados para medir a latência da melhor rota atual. Agentes exploradores tentam descobrir caminhos alternativos mais eficientes, e seu caminho é influenciado por medidas de congestionamento. Quando um agente de qualquer tipo descobre um *loop*, ou se o caminho é mais longo que o valor de MH (um parâmetro definido pelo algoritmo que define o máximo número de saltos permitidos), ele se auto-destrói. TB gera agentes de melhor rota com uma probabilidade γ e agentes exploradores com uma probabilidade $1 - \gamma$. É importante manter γ razoavelmente alta para que as medidas do melhor caminho sejam frequentes, e, assim, a descoberta de congestionamentos e a mudança para caminhos menos congestionados sejam rápidas.

Todos os agentes são inicializados pela fonte configurando-se os campos $srcAddr$ e $destAddr$, que correspondem aos endereços dos vértices fonte e destino, e configurando ainda $t_d = 0$, $hops = 0$ e $vNodes[0].address = srcAddr$, onde essas variáveis fazem parte da estrutura do agente, e são explicadas na tabela 5.1.

5.1.3 Roteamento dos Agentes

Agentes de mão única carregam consigo a latência t_d do vértice atual de volta para a fonte. Eles atualizam suas latências acumuladas em cada um dos vértices intermediários por $t_d = t_d + t(i)$, onde $t(i)$ é a atual latência do enlace de saída, que é a soma do atraso da fila e do atraso do

Tabela 5.1 - Estrutura dos Agentes do Trail Blazer. Assim com no AntNet, tanto os agentes em avanço quanto os agentes em retorno utilizam a mesma estrutura básica, mas ao contrário do AntNet existem três tipos de agentes: os agentes exploradores, os agentes de melhor rota e os agentes em retorno.

Propriedade	Descrição
<i>srcAddr</i>	endereço da fonte do agente
<i>destAddr</i>	endereço do destino do agente
<i>type</i>	tipo de agente
<i>isTwoWay</i>	TRUE: é um agente de mão dupla FALSE: é um agente de mão única
<i>hops</i>	contagem de saltos desde a fonte
<i>t_d</i>	latência do vértice atual até o destino.
<i>arrivalTime</i>	tempo de chegada ao destino.
<i>hops</i>	contagem de saltos desde a fonte
<i>vNodes</i>	uma matriz com hops + 1 entradas, uma para cada salto desde a fonte.
<i>vNodes[i].address</i>	endereço do vértice i
<i>vNodes[i].timestamp</i>	somente para agentes de mão dupla: registro de tempo na direção de vinda.

enlace. Isso não inclui o tempo de transferência, que é uma função do tamanho do pacote e largura de banda do enlace. O enlace i é o enlace na qual o agente chega do vértice anterior. $t(i)$ é calculado utilizando o mesmo modelo usado pelo AntNet-FA:

$$t(i) = d_l + \frac{q_l + s_a}{B_l} \quad (5.4)$$

onde q_l é o atual tamanho da fila do enlace i em bytes, s_a é o tamanho restante do pacote atualmente transmitido no enlace i , B_l é largura de banda do enlace i em bytes/s, e d_l é o atraso de transmissão do enlace i .

Cada vértice intermediário incrementa o contador de saltos $nhops$, e adiciona uma entrada à matriz $vNodes$. Essa entrada consiste do endereço do vértice (agentes de mão única), ou o endereço do vértice e o registro de tempo local (agentes de mão dupla).

Em cada vértice intermediário, TB calcula o conjunto S dos enlaces de saída que *não* estão conectados com os vértices que aparecem no caminho do agente. Se S for vazio, o agente é destruído.

TB roteia os agentes de melhor rota do destino d no enlace i se e somente se $i \in \mathcal{S} \wedge H(d)$, utilizando a probabilidade $p_{best}(i, d)$.

$$p_{best}(i, d) = \frac{prt_k[i, d]}{\sum_{i \in \mathcal{S}} prt_k[i, d]} \quad (5.5)$$

O agente é destruído caso $\mathcal{S} \wedge H(d)$ seja um conjunto vazio.

Ao contrário dos agentes de melhor rota, que utilizam somente a informação em prt_k e $\mathcal{S}$ para as decisões de roteamento, agentes exploradores também consideram o congestionamento atual dos enlaces de saída. O principal motivo para isso é que os agentes exploradores precisam perceber condições de congestionamento o mais cedo possível para melhor explorar caminhos alternativos. Para o caso dos agentes de mão única, foi assumido que o tráfego nos enlaces é simétrico (a mesma quantidade de fluxo de tráfego atravessa o enlace em ambas as direções), o que é normalmente incorreto. Caso o tráfego seja simétrico, um alto valor de $t(i)$ em uma direção do enlace é uma provável indicação de um congestionamento na outra direção também. Como os agentes de mão dupla medem a latência do enlace na mesma direção que os pacotes de dados, eles gerenciam o tráfego assimétrico corretamente.

TB calcula um termo de correção de congestionamento $c(i)$ para cada enlace de saída i do vértice atual. O termo de correção de congestionamento é um valor no intervalo de $[0, 1]$ que indica o congestionamento do enlace i em relação a outros enlaces, e possui uma função semelhante ao fator de correção heurístico l_n do AntNet.

$$c(i) = \frac{t(i)}{\sum_{j=1}^k t(j)} \quad (5.6)$$

onde $t(i)$ é a latência do enlace de saída i , e o vértice possui k enlaces de saída. É fácil perceber que:

$$\sum_{i=1}^k c(i) = 1 \quad (5.7)$$

TB decide enviar agentes exploradores em um enlace de saída aleatório com uma probabilidade η . A probabilidade de se selecionar um determinado enlace de saída i , $i \in \mathcal{S}$, é

$$p_{expRand}(i, d) = \frac{\frac{1}{|\mathcal{S}|} + \theta(1 - c(i))}{1 + \theta(k - 1)} \quad (5.8)$$

θ é o peso do termo de correção de congestionamento. Grandes valores de θ dão preferência para enlaces de saída com o menor fator de congestionamento atual. $\theta = 0$ força uma seleção uniforme entre os enlaces de saída sem nenhuma consideração em relação ao congestionamento. $\theta = 0.5$ é um compromisso entre a seleção uniforme e o congestionamento atual.

TB decide-se por enviar agentes exploradores utilizando a informação do melhor caminho com uma probabilidade $1 - \theta$. A probabilidade de selecionar o enlace de saída i , $i \in \mathcal{S}$, é

$$p_{expBest}(i, d) = \frac{p_{best}(i, d) + \theta(1 - c(i))}{1 + \theta(k - 1)} \quad (5.9)$$

Novamente, θ é o peso do fator de correção de congestionamento. $p_{best}(i, d)$ é a probabilidade de se enviar um agente de melhor rota no enlace de saída i . TB seleciona somente enlaces com um alto valor de $p_{best}(i, d)$ que também pertencem a $\mathcal{S}$ utilizando o algoritmo de seleção de enlaces da seção 5.1.3. Caso nenhum dos enlaces com alta probabilidade seja incluído em $\mathcal{S}$, então TB seleciona um enlace aleatório utilizando $p_{expRand}(i, d)$. Configurou-se $\eta = 0.5$ para se ter chances iguais de se selecionar um enlace de saída aleatório e um enlace de melhor caminho.

5.1.4 Roteamento dos Pacotes de Dados

Para evitar o envio de pacotes em enlaces com probabilidade baixa, o TB seleciona um conjunto $H(d)$ de enlaces de saída para cada destino d , e envia os pacotes em um dos enlaces de $H(d)$. O conjunto $H(d)$ é selecionado de tal forma que pequenas mudanças em p_{rt_k} não alterem $H(d)$. Para se calcular $H(d)$, primeiro se calcula $high(d)$, que é a maior probabilidade de qualquer enlace que leve a d .

$$high(d) = \max_{1 \leq i \leq k} p_{rt_k}[i, d] \quad (5.10)$$

Posteriormente, se calcula $low(d)$, que corresponde à menor probabilidade de qualquer enlace que seja maior que α_1 vezes $high(d)$, ou seja

$$low(d) = \min p_{rt_k}[i, d]; \quad p_{rt_k}[i, d] \geq \alpha_1 high(d) \quad (5.11)$$

Dessa forma, o conjunto $H(d)$ é calculado como:

$$H(d) = \{i | prt_k[d, i] \geq \min(\alpha_1 high(d), \alpha_2 low(d))\} \quad (5.12)$$

Dessa maneira, as constantes α_1 e α_2 implementam uma “lógica fuzzy”, que inclui enlaces com probabilidades próximas de $low(d)$.

5.1.5 Atualizando a Tabela de Roteamento

Agentes de mão única utilizam a latência acumulada t_d para atualizar a tabela prt_k no seu caminho da fonte até o destino. Agentes de mão dupla calculam o t_d no caminho de retorno, do destino até a fonte a partir dos valores calculados pelo modelo determinado para o tempo de transferência do pacote definido na seção 5.1.3. Tanto para agentes de mão única quanto para agentes de mão dupla, o vértice atualiza a tabela de latência média $avg_k(d)$.

Quando um agente atualiza $prt_k(i, d)$, isso refere-se ao enlace i , que é o enlace através do qual o agente chega ao vértice. Esse enlace aponta para o destino dos correspondentes pacotes de dados. Perceba que agentes de mão única são enviados do destinatário do fluxo de dados, e agentes de mão dupla são enviados da fonte do fluxo de dados, mas eles atualizam prt_k quando retornam do destino.

Todos os agentes atualizam prt_k utilizando o valor de t_d e o atual valor de $avg_k(d)$ a partir de:

$$f(t_d) = \max\left(\min\left(\frac{avg_k(d)}{t_d}, 10\right), 0.1\right) \quad (5.13)$$

$$\Delta p = \delta f(t_d) \quad (5.14)$$

$$prt_k[i, d] = \frac{prt_k[i, d] + \Delta p}{1 + \Delta p} \quad (5.15)$$

$$prt_k[j, d]_{j \neq i} = \frac{prt_k[j, d]}{1 + \Delta p} \quad (5.16)$$

A latência média $avg_k(d)$ é utilizada para escalar o valor de reforço positivo do agente. Um grandevvalor de $f(t_d)$ indica um melhor caminho. $f(t_d)$ é limitado ao intervalo de $]0.1, 10[$ para prevenir grandes flutuações em Δp , que é o valor de reforço de $prt_k[j, d]$.

δ é a taxa de aprendizado do algoritmo. Ela deve ser ajustada para um valor alto o suficiente para que os agentes que descubrem um melhor caminho para o destino tenham alguma influência sobre p_{rt_k} . Entretanto, ela não deve ser ajustada com um valor muito alto para que se evite oscilações. Todas as entradas em p_{rt_k} do mesmo destino d são escaladas por $1 + \Delta p$ para se garantir que a soma permaneça 1.

5.1.6 Resumo dos Parâmetros

Dessa maneira, da mesma forma que para o AntNet, pode-se resumir os parâmetros utilizados pelo Trail Blazer na tabela 5.2.

Tabela 5.2 - Sumário dos parâmetros utilizados no Trail Blazer.

Parâmetro		Valor	Descrição	Seção
M	$M > 1$	10	Número de valores para se calcular avg_k .	5.1.1
β	$\beta \in [0, 1]$	0.5	Controle a memória para mudanças de topologia.	5.1.1.1
N		50	Número de pacotes para se enviar um agente.	5.1.2
γ	$\gamma \in [0, 1]$	0.35	Probabilidade de enviar agentes de melhor rota.	5.1.2
MH		32	Máximo número de saltos do caminho.	5.1.2
η	$\eta \in [0, 1]$	0.1	Probabilidade de um agente explorador selecionar um enlace de saída aleatório.	5.1.3
θ	$\theta \in [0, 1]$	0.5	Peso do termo de correção de congestionamento.	5.1.3
α_1	$\alpha_1 \in [0, 1]$	0.5	Limite inferior para a probabilidade da seleção de enlaces no primeiro cálculo.	5.1.4
α_2	$\alpha_2 \in [0, 1]$	0.8	Limite inferior para a probabilidade da seleção de enlaces no segundo cálculo.	5.1.4
δ	$\delta \in [0, 1]$	0.2	Taxa de aprendizado.	5.1.6

5.2 Algoritmo Trail Blazer Simple

O algoritmo Trail Blazer Simple (TBs) é uma proposta de algoritmo realizada neste trabalho *a priori* de qualquer resultado obtido. Ele é obtido a partir de uma modificação realizada no TB. Essa modificação consiste de uma simplificação do processo de roteamento de dados. No caso do TBs, não é calculado o conjunto de enlaces $H(d)$ sobre o qual os pacotes de dados são encaminhados, apresentado na seção 5.1.4. Para realizar o encaminhamento dos

dados é utilizada a tabela de roteamento diretamente, da mesma maneira que o AN e ANFA. Essa modificação foi realizada visando diminuir a complexidade computacional do algoritmo e verificar a diferença nos resultados obtidos com o TB.

5.3 Algoritmo Trail Blazer Uniform

Assim como o TBs, o Trail Blazer Uniform (TBU) é outro algoritmo proposto neste trabalho. Entretanto, diferentemente do TBs, o TBU foi proposto *a posteriori* dos resultados obtidos com os outros algoritmos, na tentativa de se obter um resultado superior e corrigir determinados problemas apresentados por eles (descritos no capítulo 7), mas de tal forma que o algoritmo fosse simples e robusto, com uma menor complexidade computacional que os outros algoritmos.

Basicamente, o TBU consiste de uma modificação do TBs na tentativa de se melhorar a sua política de lançamento de agentes, mudando a política original apresentada na seção 5.1.2, por uma política equivalente às do AN e do ANFA, apresentada nas seções 4.1.2.1 e 4.1.2.2, ainda mantendo os dois tipos de agentes do TBs: os agentes exploradores e os agentes de melhor rota. Com esta modificação, é possível obter melhoras na porcentagem de pacotes entregues e no atraso dos pacotes, como é demonstrado no capítulo 7.

Capítulo 6

Modelos e Implementação

6.1 Modelo da Rede de Comunicações

As atuais redes de comunicação possuem diversas propriedades que as tornam naturalmente difíceis de se caracterizar, e portanto, de se simular [20]. Primeiro, um dos grandes motivos para o seu sucesso foi a utilização da arquitetura do Protocolo Internet (IP) para a unificação de diversas tecnologias de rede e domínios administrativos. O IP permite a inter-operação de redes significativamente diferentes. O fato do IP “mascarar” as diferenças entre as tecnologias da perspectiva do usuário não faz com que elas desapareçam. O IP consegue uma conectividade uniforme em face da diversidade, mas não um comportamento uniforme.

Uma segunda característica é que as redes de comunicação são, em geral, grandes. Isso traz dois problemas: a heterogeneidade dessas redes e a dimensão. Nem sempre existe um acordo sobre quais as características necessárias que devem ser embutidas em um modelo de simulação para redes. Assim, apresenta-se uma descrição resumida das características principais do modelo de simulação adotado.

6.1.1 Topologia

Para realizar as simulações necessárias, é necessário definir uma topologia (ou conjunto de topologias) que represente o problema a ser estudado. Para este trabalho, decidiu-se utilizar a topologia da rede japonesa NTT (Nippon Telephone and Telegraph Company). Ela consiste

de uma estrutura de 57 nós ligados por 162 enlaces bi-direcionais de fibra ótica. A largura de banda dos enlaces é de 6Mbits/s, enquanto os atrasos de propagação variam de 1ms até 5ms. A rede japonesa apresenta uma topologia irregular típica dos provedores de serviços.

Figura 6.1 - Topologia da rede japonesa NTT. Cada arco representa um enlace bi-direcional. A largura de banda é de 6Mbits/sec. Atrasos de propagação variam entre 1 e 5ms.

A preferência pela rede NTT baseou-se fundamentalmente em três motivos:

1. Apesar de que a simulação em diversas topologias diferentes seja o método mais adequado, não seria possível realizar todo um conjunto de simulações em diferentes topologias em vista da capacidade computacional disponível e do tempo disponível para realização deste trabalho.
2. Acredita-se que a topologia da rede japonesa NTT seja bem representativa para o problema sendo estudado.
3. Desde a apresentação do trabalho [7], que utilizou essa topologia, ela tem sido utilizada em diversos trabalhos sobre roteamento por colônia de formigas, e tem servido como um ponto de comparação entre diferentes algoritmos.

6.1.2 Nós e Enlaces

Todos os enlaces pertencentes a um nó são vistos como uma fila, caracterizada por uma largura de banda e um atraso de propagação. Dessa forma, os pacotes enviados pela rede utilizam o valor da largura de banda e do atraso do enlace para calcular o tempo de transmissão desse pacote [12]. Ao chegar ao nó, eles são armazenados em uma fila FIFO que representa os buffers dos roteadores. Caso esses buffers estejam carregados, o pacote é descartado. Os pacotes são divididos em duas classes: os pacotes de dados e os pacotes de roteamento. Todos os pacotes de uma mesma classe possuem a mesma prioridade.

6.1.3 Padrões de Tráfego

O tráfego na rede é definido em termos de sessões abertas entre um par de diferentes aplicações situadas em nós distintos. Essas sessões são caracterizadas por três valores: o tempo entre pacotes da sessão, o tempo entre sessões e o tamanho das sessões. Esses valores não são determinísticos, mas servem para determinar as distribuições utilizadas para gerar o tráfego. Foi utilizada uma distribuição exponencial para os valores dos tempos médios entre sessões e dos tamanhos médios das sessões, onde os valores médios foram variados de acordo com as escalas das figuras apresentadas no capítulo 7. Mas em todos os casos utilizou-se uma distribuição esponencial com média 0.05 para se calcular o tempo entre pacotes.

6.2 Simulação de Redes de Comunicação

Para realizar a simulação e obter os resultados desejados para os diferentes algoritmos foi desenvolvido um simulador de redes que permitia o roteamento de mensagens em C++. Foi utilizada uma biblioteca específica: a OMNeT++, que permite o desenvolvimento de sistemas de eventos discretos.

6.2.1 Sistemas de Eventos Discretos

Um sistema de eventos discretos é um sistema onde as mudanças de estados (eventos) acontecem em instantes de tempos discretos, e o evento leva um tempo igual a zero para acontecer, isto é, não existe uma mudança de estado no sistema entre eventos. Redes de computadores são normalmente modeladas como sistemas de eventos discretos. Por exemplo, podem-se modelar os seguintes eventos em uma rede de computadores: o começo da transmissão de um pacote, o fim da transmissão de um pacote, entre outros.

6.2.2 OMNeT++

OMNeT++ é a biblioteca de programação utilizada para simular o ambiente de redes necessário. De acordo com [12], o OMNeT é um simulador de redes de eventos discretos modular orientado a objetos. Ele consiste de um conjunto de módulos organizados hierarquicamente, cujos módulos comunicam-se através da troca de mensagens. As mensagens podem conter estruturas de dados complexas e arbitrárias. Os módulos podem mandar mensagens diretamente

para o seu destino ou através de caminhos definidos, utilizando portas e conexões.

6.2.3 Módulos Hierárquicos

Um modelo do OMNeT++ consiste de módulos organizados em uma estrutura hierárquica, que se comunicam através da passagem de mensagens. Os modelos são normalmente referidos como “rede”. O módulo mais alto da hierarquia é chamado de módulo do sistema. O módulo do sistema possui submódulos, que por sua vez também podem possuir outros submódulos. Essa estrutura de módulos é descrita em uma linguagem própria da biblioteca, a linguagem NED.

Módulos contendo submódulos são chamados módulos compostos, ao contrário dos módulos simples que são o nível mais baixo da hierarquia de módulos. Os módulos simples possuem o algoritmo do modelo. O usuário programa os módulos simples em C++, utilizando a biblioteca de classes do OMNeT++.

Figura 6.2 - Dois submódulos do OMNeT demonstrando como eles se interligam hierarquicamente. (a) Dois submódulos conectados um ao outro. (b) dois submódulos conectados ao módulo pai.

6.2.4 Mensagens, Portas e Enlaces

Os módulos comunicam-se através da troca de mensagens. Na simulação, mensagens podem representar pacotes em uma rede de computadores, trabalhos ou clientes em uma fila ou outro tipo de entidades móveis. As mensagens podem possuir estruturas de dados complexas, de acordo com a necessidade. Módulos simples podem enviar mensagens diretamente para algum destino ou através de um caminho definido, por meio de portas e conexões.

O “tempo de simulação local” do módulo avança quando ele recebe uma mensagem. A mensagem pode chegar de um outro módulo ou dele próprio (as mensagens próprias são utilizadas para implementar alarmes).

Portas são as interfaces de entrada e saída dos módulos, mensagens são enviadas através das portas de saída e recebidas através das portas de entrada. Cada conexão (também chamada de enlace) é criada em um único nível na hierarquia de módulos: dentro de um módulo composto é possível conectar as portas correspondentes de dois submódulos.

6.2.5 Transmissão de Mensagens

Conexões podem possuir três parâmetros, que facilitam a modelagem de redes de comunicação, mas que podem ser úteis em outros modelos também:

Atraso de propagação é a quantidade de tempo que as mensagens gastam para atravessar um canal.

Taxa de erro de bits especifica a probabilidade que um bit possui de ser transmitido incorretamente, e permite a modelagem simples de um canal com ruído.

Taxa de transmissão de dados é especificada em bits/s, e é utilizada para calcular o tempo de transmissão da mensagem.

Quando a taxa de transmissão é utilizada, o envio de mensagens no modelo corresponde à transmissão do primeiro bit, e a chegada da mensagem corresponde à recepção do último bit. Esse modelo não é sempre aplicável, por exemplo, protocolos tipo Token Ring ou FDDI não esperam para receber o pacote inteiro, mas começam a repetir o primeiro bit logo após recebê-lo.

6.3 Implementação do Simulador

Esta seção apresenta a estrutura básica do simulador implementado em C++. O código fonte completo do simulador para os diversos algoritmos encontra-se no CD acompanhante da dissertação. Dessa forma, evita-se a apresentação de uma grande quantidade de código que em um primeiro momento não é muito útil para se compreender o programa. Apresenta-se, entretanto, uma explicação da estrutura do programa, com relação aos seus arquivos fontes, que permite uma melhor compreensão do seu funcionamento, e relacionamento com os algoritmos previamente apresentados.

6.3.1 Estrutura Geral do Simulador

Utilizando as idéias de programação do OMNeT++ e do trabalho [14], o simulador foi dividido em diversos módulos menores, responsáveis por funções mais específicas (uma abordagem dividir para conquistar, típica de programação). Dividiu-se o programa nos seguintes módulos:

1. Gerador de Agentes
2. Sorvedouro de Agentes
3. Ninho dos Agentes
4. Gerador de Dados
5. Sorvedouro de Dados
6. Roteador Interno
7. Topologia

Todos esses módulos são explicados nesta seção. Apesar disso, é possível verificar a interligação desses módulos na Figura 6.3, que apresenta o funcionamento lógico do simulador. Todos as caixas cinza claro representam módulos independentes que se comunicam através de mensagens, representadas pelas setas pretas. A direção da seta indica a direção da comunicação. Ao lado dessas setas existe uma pequena legenda que indica o tipo de mensagem transmitida. Dentro de cada módulo existe uma descrição de alto nível explicando a função do módulo e uma demonstração gráfica das suas estruturas de dados. Dessa forma, é fácil perceber que o ninho dos agentes possui uma estrutura de dados para armazenar o modelo de tráfego necessário para os algoritmos, enquanto o roteador interno possui uma tabela de roteamento necessária para o encaminhamento de pacotes. É possível também ver que o ninho de formigas têm a capacidade de acessar e modificar a tabela de roteamento do roteador interno.

Vale ressaltar ainda que o diagrama da Figura 6.3 apresenta um único roteador da rede. Para o caso da rede japonesa NTT, existem 57 roteadores como esse que se comunicam entre si, trocando mensagens de dados e de agentes.

6.3.2 Pacotes de Dados e de Agentes

A comunicação entre os diferentes módulos do simulador é realizada basicamente através de dois tipos de mensagens: os pacotes de dados e os pacotes de agentes. Ambos os tipos de mensagem derivam da classe base `cMessage` implementada pelo OMNeT++. Os pacotes de dados representam os dados que são transmitidos na rede entre os diferentes roteadores da rede, e

Figura 6.3 - Estrutura Básica dos módulos do simulador. Cada caixa cinza representa um módulo separado do simulador. As linhas com setas entre os módulos representam linhas de comunicação entre os módulos (não representam necessariamente enlaces, apesar de enlaces serem representados por elas).

<i>samplePacket</i>	
<i>parâmetros:</i> id sourceAddress destAddress hops ...	<i>métodos:</i>

Figura 6.4 - Descrição do módulo de mensagem de dados.

<i>ant</i>	
<i>parâmetros:</i> sourceNode destNode hops visitedNodes[] antStackEntry cycleNodes[]	<i>métodos públicos:</i> setVisitedNodesArraySize getVisitedNodesArraySize getVisitedNodes setVisitedNodes addToVisitedNodes ...

Figura 6.5 - Descrição do módulo de mensagem dos agentes.

encontram-se implementados no arquivo `Messages.msg` com o nome de `samplePacket`. Além dessa estrutura, esse arquivo possui também outras duas estruturas de mensagens: `helloPacket` e `helloAndReplyMessage`, que são utilizadas somente na inicialização dos roteadores.

Os pacotes de agentes representam os agentes responsáveis pela comunicação e atualização das estruturas de dados dos roteadores, e encontram-se implementados nos arquivos `ant.msg`, `ant.cpp` e `ant.h`.

6.3.3 Gerador de Agentes

A classe geradora de agentes é responsável pela criação dos pacotes de agentes, e encontra-se implementada nos arquivos `antGenerator.ned`, `antGenerator.h` e `antGenerator.cpp`. Dependendo do algoritmo simulado, ela foi implementada de maneira diferente, de acordo com a especificação do algoritmo.

antGenerator	
<i>parâmetros:</i> iaTime (an) packetRate (TB) bestpathAgentProb (tb) address numStations messageLength ...	<i>métodos públicos:</i>
<i>portas:</i> ◀ toAntNest ▶ fromDataGen	

Figura 6.6 - Descrição do módulo de geração dos agentes.

6.3.4 Sorvedouro de Agentes

A função do sorvedouro de agentes é simular o roteador de destino que é o destino do agente e coletar os dados armazenados pelo agente para a análise do algoritmo. Ele encontra-se implementado nos arquivos `antSink.ned`, `antSink.h` e `antSink.cpp`.

6.3.5 Ninho dos Agentes

O ninho dos agentes é onde realmente o processamento é realizado e é onde a maior parte dos algoritmos é implementada. Ele encontra-se nos arquivos `antNest.ned`, `antNest.h`, `antNet.cpp`. A principal função do ninho de agentes é receber os agentes do roteador interno e atualizar a sua tabela de roteamento. Caso o agente sendo analisado tenha como destino o roteador atual, o ninho envia esse agente para o sorvedouro de agentes, caso contrário, ele insere uma entrada na memória do agente e calcula o próximo salto no caminho do agente. Então, ele devolve o agente para o roteador interno, que irá encaminhar o agente para o seu próximo salto, até que o agente chegue ao seu destino ou seja destruído.

A estrutura do ninho é bastante dependente do algoritmo que ele representa, mas mesmo assim, existem certas características comuns a todos os algoritmos. Nas figuras 6.7 e 6.8 apresentam-se a estrutura básica para o ninho do AN e para o ninho do TB.

antNest (AN)	
<i>parâmetros:</i> expMeanCoefficient windowSizeCoefficient zetaConfidenceLevel explorationProbablity rescalingPower queueWeight forkProbability squashFunctionCoefficient maxHopsCoefficient ageLimit timeWeight probabilisticRouting sleepTimeAtStart converganceTime ...	<i>métodos privados:</i> processForwardAnts processBackwardAnts selectLinks updateRoutingTable... heuristicCorrectionFactor getReinforcement squashingFunction squashedReinforcement ...
<i>portas:</i> ◀ toAntSink ◀ toRouter ▶ fromRouter ▶ fromAntGen	

Figura 6.7 - Descrição do módulo de ninho dos agentes do AN e ANFA.

6.3.6 Gerador de Dados

A função do gerador de dados é simular o computador fonte que é quem gera os dados para a análise do algoritmo. Esse módulo gera os dados assim como descrito na seção 6.1.3. Ele encontra-se implementado nos arquivos `dataGen.ned`, `dataGen.h` e `dataGen.cpp`.

6.3.7 Sorvedouro de Dados

A função do sorvedouro de dados é simular o computador de destino que é o destino dos dados e coletar os dados estatísticos para a análise do algoritmo. Ele encontra-se implementado nos arquivos `dataSink.ned`, `dataSink.h` e `dataSink.cpp`.

6.3.8 Fila do Roteador

A fila do roteador representa a fila de pacotes armazenada no roteador esperando para ser encaminhada para os seus respectivos destinos. Ao contrário dos módulos anteriores, a fila

antNest (TB)	
<i>parâmetros:</i> learningRate expMeanCoefficient windowSizeCoefficient zetaConfidenceLevel explorationProbablity randomExplorationProbablity congestionWeigth rescalingPower queueWeight forkProbability squashFunctionCoefficient maxHopsCoefficient ageLimit timeWeight probabilisticRouting sleepTimeAtStart converganceTime ...	<i>métodos privados:</i> processForwardAnts processBackwardAnts selectLinks updateRoutingTable... calculateProbBest calculateProbExpRand calculateProbExpBest ...
<i>portas:</i> ◀ toAntSink ◀ toRouter ▶ fromRouter ▶ fromAntGen	

Figura 6.8 - Descrição do módulo de ninho dos agentes do TB, TBs e TBu.

do roteador não é um módulo, mas somente um objeto utilizado por um módulo (o roteador interno). Ela é implementada nos arquivos `buffer.h` e `buffer.cpp`.

Ao contrário do que o nome indica, ela pode ser imaginada como duas filas independentes: a fila normal e a fila prioritária. A fila normal é utilizada pelos pacotes de dados e pelos agentes em avanço do AN. A fila prioritária, como o nome indica, possui prioridade sobre a fila normal. Ela é utilizada pelos agentes em avanço do AN e pelos demais agentes dos outros algoritmos. Esse funcionamento pode ser visualizado na figura 6.3.

Ambas as filas são programadas como um modelo FIFO, utilizando a classe `vector<T>` da Standard Template Library. Dessa forma, os pacotes que chegam antes saem antes, desde que possuam uma mesma prioridade.

Figura 6.9 - Descrição do módulo de roteador interno.

6.3.9 Tabela de Roteamento

Assim como a fila do roteador, a tabela de roteamento não é um módulo, mas uma classe utilizada pelo módulo roteador interno. Essa classe tem a função de representar uma tabela de roteamento que é utilizada pelo roteador interno para encaminhar os pacotes de dados recebidos. Ela armazena a tripla de informações sobre destino, vizinho e probabilidade, necessárias para o encaminhamento. A tabela de roteamento é implementada nos arquivos fontes `routingTable.h` e `routingTable.cpp`.

6.3.10 Roteador Interno

O roteador interno é o módulo que implementa a funcionalidade da camada de roteamento da rede. Ele é responsável por receber os pacotes de dados vindos do módulo de geração de dados ou de outros roteadores e encaminhar esses dados para outros roteadores ou para o servidor de dados, caso seja o endereço de destino do pacote. Ele é implementado nos arquivos `genericRouter.ned`, `router.h` e `router.cpp`, e apesar das diferenças de algoritmos, ele é basicamente o mesmo para os diferentes algoritmos, sendo mostrado na Figura 6.9.

Capítulo 7

Resultados Obtidos e Discussão

Neste capítulo são apresentados os resultados obtidos pelo simulador desenvolvido em C++ e explicado no capítulo anterior, para os diversos algoritmos analisados utilizando a topologia da rede japonesa NTT. As tabelas com os resultados completos das simulações para todos os casos analisados encontram-se disponíveis no Anexo A. Não foram simuladas falhas de enlaces em nenhum dos casos estudados.

Foram utilizadas diversas métricas padrão:

Pacotes entregues é a média do valor dos pacotes roteados corretamente durante o período de simulação. É normalmente apresentado como a porcentagem entre o total de mensagens geradas e o total de mensagens entregues pela rede.

Atraso dos pacotes mede o tempo de viagem dos pacotes e a variância desse valor. Um outro dado importante é o atraso dos 90% dos pacotes que corresponde a um intervalo de tempo máximo no qual 90% dos pacotes foram entregues corretamente.

Largura de banda utilizada é adotada como uma medida da utilização dos recursos da rede, sendo apresentada como uma porcentagem da largura de banda total da rede, durante o tempo de simulação.

Número de saltos mede o número de saltos médios (o número de roteadores pelos quais o pacote teve que passar) para chegar ao computador destino.

Número de agentes gerados representa o número de agentes (pacotes de controle) gerados pelos diferentes algoritmos.

Número de agentes recebidos representa o número de agentes recebidos corretamente

pelos diferentes algoritmos, sendo normalmente apresentado como uma porcentagem entre o total de agentes gerados em relação ao total de agentes recebidos.

Tempo de latência dos agentes mede o tempo de viagem dos agentes, assim como para o atraso dos pacotes.

Overhead dos agentes mede o overhead da utilização por parte dos agentes em relação à largura de banda total da rede durante o tempo de simulação.

Para avaliar o desempenho dos diferentes algoritmos, todos os resultados são comparados com uma implementação simplificada do OSPF, onde as tabelas de roteamento foram ajustadas pelo resultado da computação do caminho mais curto. As simulações foram realizadas para um período de 1000s, fornecendo antes um período de 50s para convergência, na qual permitiu-se que os agentes fossem gerados, mas nenhum pacote de dado.

7.1 Porcentagem dos Pacotes Entregues

A porcentagem de pacotes entregues é uma estatística chave para se analisar a performance de uma rede de comunicações. Ela é medida pela relação entre os pacotes colocados na rede pelo gerador de dados e os pacotes efetivamente entregues no seu destino, recebidos pelo sorvedouro de dados.

$$p.p.e. = 100 \times \frac{\text{pacotes recebidos}}{\text{pacotes gerados}} \quad (7.1)$$

As figura 7.1 e 7.2 fornecem os resultados obtidos para a porcentagem de pacotes entregues para os diferentes algoritmos analisados em suas configurações padrão e para os diferentes parâmetros de tráfego gerado. Na figura 7.2, todos os gráficos encontram-se sumarizados com linhas de contorno correspondentes às taxas de entrega de pacotes do OSPF, pelas quais os resultados foram sumarizados na figura 7.1, o que facilita a comparação entre os diferentes algoritmos.

Um outro gráfico apresentado em conjunto é o ganho do algoritmo em relação ao OSPF, que para o caso da porcentagem de pacotes entregue é definido como:

$$Ganho(p.p.e.) = 100 \times \frac{p.p.e. \text{ do algoritmo}}{p.p.e. \text{ do OSPF}} \quad (7.2)$$

Nesse gráfico também se apresentam as linhas de contorno para as taxas de entrega de pacotes de OSPF, o que facilita a verificação de para qual valor da porcentagem de entrega de pacotes do OSPF o ganho dos diferentes algoritmos analisados é maior.

Os resultados encontrados nesses gráficos podem ser sumarizados na tabela 7.1, onde o resultado para a porcentagem de pacotes entregues pelos diferentes algoritmos foi sumarizado pela porcentagem de pacotes entregue pelo OSPF. Obviamente, isto representa a média dos valores obtidos entre as linhas das diferentes curvas de nível da figura 7.2.

Analizando esses resultados, percebe-se que para uma pequena taxa de perda de pacotes ($<2\%$) do OSPF, todos os algoritmos possuem um desempenho similar. Com o aumento do tráfego na rede, os resultados mostram que os algoritmos baseados em agentes possuem um desempenho superior ao do OSPF na entrega de pacotes, conseguindo entregar a maior parte dos pacotes de dados, até onde existem altas taxas de perdas do OSPF. Os algoritmos baseados em agentes possuem o seu maior ganho no intervalo de taxa de perdas de pacotes entre 25% e 35% do OSPF, o que pode ser verificado analisando-se as figuras de ganho da figura 7.1, que mostram o ganho dos algoritmos baseados em agentes em relação ao OSPF. Em alguns casos o ganho chega a 200%, como pode ser visto na figura 7.2, enquanto a taxa de perda de pacotes é menor que 5%, de acordo com a tabela 7.1. Apesar dessas taxas de perda de pacotes serem incomuns em redes operacionais, não é incomum que certos enlaces em horário de pico de movimento cheguem a ter uma taxa de perda ao redor de 30%. Acima de 40% existe uma degradação dos algoritmos baseados em agentes por causa da grande saturação da rede, como pode ser verificado pelas figuras de ganho da figura 7.2. A partir desse ponto, o desempenho dos algoritmos baseados em agentes torna-se equivalente ao do OSPF.

Pelos resultados apresentados, também percebe-se que o AN, ANFA e TBU possuem um desempenho levemente superior ao do TB e do TBs, apesar de que a diferença é mínima e pode ser facilmente desprezada. Uma outra observação é que as diferenças entre o TB e o TBs e entre o AN e o ANFA também são mínimas. Esperava-se que o ANFA tivesse um desempenho superior ao do AN, o que não ocorreu.

Tabela 7.1 - Resultados para a porcentagem de pacotes entregues para os diferentes algoritmos de roteamento.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	99.426%	99.963%	99.953%	99.957%	99.971%	99.995%
80-90	83.442%	99.964%	99.963%	99.051%	99.995%	99.984%
70-80	71.855%	99.969%	99.969%	97.468%	97.546%	99.881%
60-70	64.882%	95.265%	93.737%	91.136%	90.508%	94.586%
50-60	53.384%	66.666%	64.026%	65.919%	66.429%	66.287%
40-50	42.659%	48.003%	48.219%	50.574%	49.303%	49.366%
30-40	33.547%	34.766%	35.614%	36.772%	36.837%	36.408%
20-30	24.029%	21.580%	22.819%	26.068%	24.762%	24.550%
10-20	17.403%	12.892%	14.737%	17.404%	16.217%	16.119%
0-10	9.796%	4.935%	6.138%	8.259%	7.562%	7.320%

Figura 7.1 - Resultados para a porcentagem de pacotes entregues para os diferentes algoritmos de roteamento.

(a)

(b)

(c)

(d)

(e)

Figura 7.2 - Dados obtidos para a porcentagem de pacotes entregues em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(a) OSPF

(b) AN

(c) Ganho do AN

(d) ANFA

(e) Ganho do ANFA

(f)

(g)

(h)

(i)

(j)

(k)

Continuação Figura 7.2 - Dados obtidos para a porcentagem de pacotes entregues em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(f) TB

(g) Ganho do TB

(h) TBs

(i) Ganho do TBs

(j) TBu

(k) Ganho do TBu

7.2 Atraso dos Pacotes

Como descrito no começo deste capítulo, o atraso dos pacotes é medido utilizando-se três métricas diferentes: a média do atraso dos pacotes, o desvio padrão do atraso e o intervalo do atraso máximo para 90% dos pacotes. O atraso mede o tempo de viagem para os pacotes de dados entregues corretamente no seu destino. O intervalo do atraso máximo para 90% dos pacotes é calculado a partir da média e do desvio padrão como:

$$90delay = \mu + 1,29\sigma \quad (7.3)$$

onde μ é o atraso médio e σ é o desvio padrão do atraso. Esta equação pode ser comprovada assumindo-se que o tempo de atraso dos pacotes possua uma distribuição gaussiana, o que é uma suposição razoável. Dessa forma deseja-se que a área sobre a função densidade de probabilidade seja igual a 0,9. Isso corresponde a que a área restante seja igual a 0,1. Utilizando a tabela da referência [21], chega-se ao valor aproximado de 1,29.

Assim, os três valores obtidos pelos resultados da simulação para os diferentes algoritmos nas suas configurações padrões, o atraso médio, o desvio padrão e o atraso médio para 90% dos pacotes são apresentados nas tabelas 7.2, 7.3 e 7.4, respectivamente. Podem-se ver também os resultados do intervalo do atraso máximo para 90% dos pacotes para as diferentes condições de tráfego na figura 7.3.

Analisando-se esses resultados é fácil perceber as grandes diferenças entre os resultados obtidos para o comportamento do AN, ANFA e TBu em relação ao TB e TBs, em suas configurações padrão. O desempenho do AN, ANFA e TBu é muito superior, tendo uma grande vantagem sobre o OSPF, TB e TBs.

Pode-se verificar que para baixo tráfego (taxa de perda $< 2\%$ para o OSPF), os algoritmos baseados em agentes possuem o menor tempo de atraso médio, conseguindo manter ainda um desvio padrão do atraso muito menor que o OSPF, chegando a ser 10 vezes menor que o do OSPF. Dessa forma, o atraso máximo para 90% dos pacotes também é menor, o que para aplicações multimídia é um grande ganho. Essa condição se mantém até taxas de perda de pacotes da ordem de 40% para o OSPF, condição na qual o atraso dos algoritmos baseados em agente torna-se equivalente ao OSPF.

Além dessa diferença entre os algoritmos baseados em agentes e o OSPF, também existem uma grande diferença entre o AN, ANFA e TBu e o TB e TBs. É possível ver pelas tabelas e pelo atraso porcentual na figura 7.6, que os algoritmos AN, ANFA e TBu conseguem um resultado muito melhor que o TB e TBs, tendo um desempenho médio 50% superior a estes dois últimos algoritmos para as condições de tráfego com pouco e médio tráfego.

Tabela 7.2 - Média do atraso dos pacotes de dados, em segundos.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	0.3095	0.0641	0.0636	0.1171	0.0995	0.0567
80-90	1.8846	0.0941	0.1132	0.2877	0.3218	0.1049
70-80	1.3836	0.1607	0.1570	0.6163	0.5856	0.2029
60-70	1.6201	1.4214	1.5253	1.5924	1.6458	1.1117
50-60	2.0466	3.0423	2.6580	2.7003	2.7267	2.6460
40-50	2.5788	3.5163	3.4781	3.0929	3.2241	3.2650
30-40	3.0617	3.9515	3.9869	3.5229	3.4917	3.6672
20-30	3.6772	4.6892	4.6891	3.9172	4.2525	4.2863
10-20	4.1183	5.1894	4.9644	4.5005	4.6603	4.7084
0-10	4.1727	5.9511	5.6322	5.1907	5.2459	5.3563

Figura 7.3 - Média do atraso dos pacotes de dados, em segundos.

Tabela 7.3 - Desvio Padrão do atraso dos pacotes de dados, em segundos.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	0.5473	0.0654	0.0708	0.2472	0.2094	0.0636
80-90	3.0027	0.1379	0.2234	0.9817	0.9095	0.2510
70-80	2.7366	0.2658	0.2491	1.6064	1.5263	0.5969
60-70	2.8876	1.7363	1.8681	2.2785	2.3500	1.9142
50-60	3.0974	3.0087	3.0977	3.0608	3.0722	3.0914
40-50	3.3008	3.1418	3.3033	3.2240	3.2653	3.2642
30-40	3.4141	3.1805	3.2517	3.3077	3.3350	3.2981
20-30	3.4576	3.1588	3.2489	3.3353	3.3446	3.3096
10-20	3.4685	3.1295	3.2490	3.3317	3.3211	3.3006
0-10	3.4738	2.6047	2.8386	3.0965	3.0702	3.0018

Figura 7.4 - Desvio Padrão do atraso dos pacotes de dados, em segundos.

Tabela 7.4 - Intervalo máximo de atraso para 90% dos pacotes, em segundos.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	1.0156	0.1484	0.1549	0.4360	0.3696	0.1388
80-90	5.7580	0.2720	0.4013	1.5540	1.4951	0.4286
70-80	4.9137	0.5036	0.4784	2.6886	2.5545	0.9730
60-70	5.3451	3.6613	3.9351	4.5316	4.6773	3.5811
50-60	6.0423	6.9235	6.6540	6.6487	6.6898	6.6339
40-50	6.8369	7.5692	7.7394	7.2518	7.4363	7.4758
30-40	7.4658	8.0543	8.1816	7.7898	7.7938	7.9217
20-30	8.1375	8.7640	8.8801	8.2198	8.5671	8.5557
10-20	8.5926	9.2266	9.1556	8.7984	8.9445	8.9661
0-10	8.6538	9.3111	9.2941	9.1852	9.2065	9.2286

Figura 7.5 - Intervalo máximo de atraso para 90% dos pacotes, em segundos.

(a)

(b)

(c)

(d)

(e)

Figura 7.6 - Resultados obtidos para o intervalo máximo do atraso de 90% dos pacotes em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(a) OSPF

(b) AN

(c) Atraso porcentual do AN em relação ao OSPF

(d) ANFA

(e) Atraso porcentual do ANFA em relação ao OSPF

(f)

(g)

(h)

(i)

(j)

(k)

Continuação Figura 7.6 - Resultados obtidos para o intervalo máximo do atraso de 90% dos pacotes em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(f) TB

(g) Atraso porcentual do TB em relação ao OSPF

(h) TBs

(i) Atraso porcentual do TBs em relação ao OSPF

(j) TBU

(k) Atraso porcentual do TBU em relação ao OSPF

7.3 Largura de Banda Utilizada

A largura de banda utilizada é uma medida do grau de utilização dos recursos da rede pelos diferentes algoritmos, sendo apresentada como uma porcentagem da largura de banda total da rede, durante o tempo de simulação. Os algoritmos baseados em agentes possuem uma habilidade natural para realizar o roteamento dos pacotes de dados por diversos caminhos, favorecendo a utilização da largura de banda da rede que de alguma forma estaria ociosa naquele instante, e permitindo um roteamento adaptativo em relação ao tráfego.

Pode-se calcular a largura de banda total da rede a partir de:

$$BW_{total} = \text{Número de enlaces} \times \text{b.w. do enlace} \quad (7.4)$$

que corresponde à soma da largura de banda de todos os enlaces de todos os roteadores da rede. Pode-se calcular a largura de banda utilizada pelo sistema a partir de:

$$BW_{utilizada} = \frac{\text{total de bits transmitidos}}{\text{tempo de simulação}} \quad (7.5)$$

onde o total de bits transmitidos é uma métrica coletada em cada um dos roteadores simulados e encontra-se nas tabelas do Anexo A com os resultados para os diferentes algoritmos considerados. Dessa forma, chega-se aos resultados mostrados na figura 7.7 e sumarizados na tabela 7.5 para a utilização da largura de banda da rede.

Analisando-se os resultados pelos intervalos da taxa de perda de pacotes nota-se facilmente que, para pequenas taxas de perda de pacotes ($< 2\%$), existe pouca diferença entre as larguras de banda utilizadas pelos algoritmos baseados em agentes e pelo OSPF. Isso obviamente ocorre por causa do pouco tráfego existente na rede gerando poucos (ou nenhum) congestionamentos nos enlaces.

Com o aumento da taxa de perda de pacotes do OSPF, é fácil verificar a maior utilização da rede pelos algoritmos baseados em agentes, o que justificaria a maior porcentagem de pacotes entregues e a menor latência observadas anteriormente. Utilizando-se dos gráficos da figura 7.7, percebe-se que os algoritmos baseados em agentes quase duplicam a largura de banda utilizada pela rede, principalmente para os caso em que as sessões são maiores e o intervalo entre sessões também é maior.

Tabela 7.5 - Largura de banda utilizada pelos diferentes algoritmos.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	11.314%	16.485%	16.001%	11.969%	12.263%	13.035%
80-90	18.728%	30.350%	29.741%	23.377%	24.504%	25.470%
70-80	20.587%	37.783%	37.111%	30.241%	30.886%	32.508%
60-70	27.575%	56.337%	56.377%	44.147%	44.919%	47.187%
50-60	39.011%	70.238%	69.806%	56.088%	57.963%	58.681%
40-50	46.396%	74.820%	76.039%	62.251%	64.003%	67.431%
30-40	54.236%	80.221%	81.292%	68.667%	70.883%	73.614%
20-30	62.568%	85.854%	85.822%	75.659%	78.257%	79.913%
10-20	68.003%	90.329%	88.883%	81.130%	83.237%	84.417%
0-10	77.185%	96.811%	94.793%	90.327%	91.840%	92.320%

Figura 7.7 - Largura de banda utilizada pelos diferentes algoritmos.

(a)

(b)

(c)

(d)

(e)

Figura 7.8 - Largura de banda utilizada em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(a) OSPF

(b) AN

(c) Largura de banda utilizada pelo AN em relação ao OSPF

(d) ANFA

(e) Largura de banda utilizada pelo ANFA em relação ao OSPF

(f)

(g)

(h)

(i)

(j)

(k)

Continuação Figura 7.8 - Largura de banda utilizada em função do intervalo da sessão e do tamanho da sessão, para os diferentes algoritmos.

(f) TB

(g) Largura de banda utilizada pelo TB em relação ao OSPF

(h) TBs

(i) Largura de banda utilizada pelo TBs em relação ao OSPF

(j) TBu

(k) Largura de banda utilizada pelo TBu em relação ao OSPF

7.4 Número de Saltos dos Pacotes de Dados

O número médio de saltos (o número médio de roteadores pelo qual o pacote teve que passar para chegar ao destino) é uma medida utilizada para verificar o grau de adaptabilidade dos diferentes algoritmos em relação às condições de tráfego. Apesar de que um maior número de saltos pode melhorar o desempenho da rede, deve-se tentar evitar um número excessivo de saltos desnecessários. Esses valores encontram-se sumarizados na tabela 7.6.

Para baixas taxas de perda de pacotes, o número médio de saltos do OSPF é quase o mesmo que o do TB, TBs e TBu, sendo que o AN e o ANFA já fazem um roteamento mais adaptativo. Com o aumento da taxa de perda de pacotes, em todos os caso existe uma maior adaptação às condições da rede pelos algoritmos baseados em agentes.

Outro detalhe é a queda do número médio de saltos dos pacotes de dados para o OSPF de acordo com o aumento da taxa de perdas de pacotes. Em um primeiro momento, seria de se esperar que esse valor fosse constante, visto que a topologia da rede não mudou, e a probabilidade de se escolher diferentes destinos continua a mesma. Entretanto, esse valor realmente cai, visto que essa estatística só é calculada para os pacotes que chegam ao seu destino. Com o aumento do tráfego na rede, a probabilidade de pacotes com destinos mais distantes chegarem ao seu destino diminui, e é mais provável que pacotes com destinos mais próximos cheguem mais facilmente, em vista do aumento do tráfego. E como somente se utilizam as estatísticas dos pacotes que chegam ao seu destino, esse valor acaba caindo com o aumento da taxa de perda de pacotes.

Ao analisar-se o número de agentes gerados pelos diferentes algoritmos, na tabela 7.7, percebe-se a diferença entre os resultados do AN, ANFA e TBu e dos resultados do TB e TBs. Percebe-se que para o AN, o ANFA e o TBu, o número de agentes lançados é praticamente o mesmo, independente da taxa de perda de pacotes. Isso favorece a que ele tenha uma ótima porcentagem de pacotes de dados entregues e um baixo tempo de latência, ambos demonstrados nas seções 7.1 e 7.2. Ao contrário, o TB e o TBs, por causa do seu método de lançamento de agentes, geram muito poucos agentes nas condições de baixo tráfego, o que os impossibilita de melhorar o desempenho de latência dos pacotes, e geram um quantidade muito grande de agentes nas condições de grande congestionamento, uma condição que não pode ser melhorada mesmo com um grande número de agentes, em vista da grande saturação da rede. Esse problema do TB e do TBs é melhor visto com os resultados apresentados na seção 7.7.

Tabela 7.6 - Número de saltos médios para os pacotes de dados para os diferentes algoritmos de roteamento.

Intervalo	OSPF	AN	ANFA	TB	sTB	TBu
90-100	6.1344	8.3613	8.0952	6.4093	6.5321	6.9052
80-90	5.5945	8.4499	8.2467	6.7233	7.1723	7.2375
70-80	4.9465	8.5442	8.4291	6.9846	7.1693	7.3632
60-70	4.6556	8.5467	8.4739	6.9218	7.0456	7.2579
50-60	4.1899	6.4303	6.2104	5.4617	5.6404	5.6789
40-50	3.7496	5.0932	5.0860	4.5726	4.6490	4.9140
30-40	3.3014	4.1043	4.0054	3.9007	3.9193	4.1427
20-30	2.8450	3.0347	3.0400	3.2238	3.1740	3.3549
10-20	2.4869	2.3078	2.4349	2.6867	2.6629	2.8147
0-10	1.9807	1.4680	1.6649	1.9315	1.8398	1.9446

Figura 7.9 - Número de saltos médios para os pacotes de dados para os diferentes algoritmos de roteamento.

7.5 Número de Agentes Gerados e Coletados

Além de se analisar o desempenho da rede em relação à taxa de perda de pacotes, para realizar uma melhor comparação dos diferentes algoritmos baseados em agentes, também foram coletadas métricas relativas aos agentes dos diferentes algoritmos nas suas configurações padrões. Estas métricas são: o número de agentes gerados, a porcentagem de agentes utilizados no reforço dos algoritmos, o tempo de vida médio dos agentes, o número de saltos médios realizados pelos agentes e a largura de banda utilizada pelos agentes. Esses valores encontram-se respectivamente nas tabelas 7.7, 7.8, 7.9, 7.10 e 7.11.

7.6 Tempo de Latência dos Agentes

O tempo de latência dos agentes mede o tempo de viagem que eles demoram a chegar ao seu destino e retornar com as informações necessárias. O sumário desse resultado encontra-se na tabela 7.9.

Com esse resultado é fácil perceber a desvantagem do AN em relação aos outros algoritmos baseados em agentes que utilizam um modelo para se calcular o tamanho da fila de espera dos pacotes de dados. Apesar desse resultado, não foram notadas diferenças significativas entre o desempenho do AN e do ANFA.

7.7 Largura de Banda Utilizada pelos Agentes

Outra diferença entre os algoritmos está no overhead de controle gerado por cada um nas suas configurações padrões. Essa métrica encontra-se sumarizada na tabela 7.11.

Existe uma grande diferença entre as larguras de banda de controle geradas pelo AN, pelo ANFA e pelo TBu e as geradas pelo TB e pelo TBs. Isso já era de se esperar ao se ver a diferença do número de agentes lançados pelos diferentes algoritmos. Dessa forma, é fácil verificar que os algoritmos com uma política de lançamento uniforme com a determinada taxa na configuração padrão utilizam uma largura de banda maior da rede. Entretanto, mesmo no pior caso apresentado, estes algoritmos utilizam no máximo 0.01% da largura de banda da rede. Esta utilização da rede é mais do que compensada pelos ganhos de desempenho obtidos.

Tabela 7.7 - Número de agentes gerados (10^5) pelos diferentes algoritmos.

Intervalo	AN	ANFA	TB	sTB	TBu
90-100	2.5838	2.5837	0.1617	0.1625	2.5837
80-90	2.5842	2.5841	0.3130	0.3079	2.5841
70-80	2.5838	2.5837	0.3956	0.3939	2.5837
60-70	2.5837	2.5838	0.6004	0.6005	2.5838
50-60	2.5838	2.5836	1.0374	1.0333	2.5836
40-50	2.5838	2.5839	1.4718	1.4773	2.5839
30-40	2.5838	2.5837	2.1174	2.1079	2.5837
20-30	2.5838	2.5839	3.0993	3.1358	2.5839
10-20	2.5837	2.5836	4.4322	4.4484	2.5836
0-10	2.5841	2.5842	8.7726	8.7658	2.5842

Figura 7.10 - Número de agentes gerados (10^5) pelos diferentes algoritmos.

Tabela 7.8 - Porcentagem de agentes utilizados no reforço que voltaram a sua fonte.

Intervalo	AN	ANFA	TB	sTB	TBu
90-100	32.116%	30.897%	94.882%	94.991%	95.207%
80-90	41.439%	40.058%	94.959%	94.816%	95.173%
70-80	46.545%	44.671%	94.754%	94.766%	94.956%
60-70	55.065%	52.908%	94.318%	94.348%	94.561%
50-60	54.090%	56.515%	94.065%	94.016%	94.125%
40-50	45.724%	58.324%	93.938%	93.930%	93.934%
30-40	35.315%	60.105%	93.926%	93.956%	93.941%
20-30	23.305%	61.809%	93.922%	93.924%	93.923%
10-20	14.442%	62.673%	93.913%	93.916%	93.914%
0-10	5.234%	65.031%	94.022%	93.985%	94.003%

Figura 7.11 - Porcentagem de agentes utilizados no reforço que voltaram a sua fonte.

Tabela 7.9 - Tempo de latência máximo para 90% dos agentes, em segundos.

Intervalo	AN	ANFA	TB	sTB	TBu
90-100	0.1844	0.1743	0.1173	0.1161	0.1005
80-90	0.2449	0.1778	0.1183	0.1217	0.1135
70-80	0.4016	0.1744	0.1232	0.1235	0.1184
60-70	4.2296	0.1752	0.1280	0.1281	0.1254
50-60	14.2910	0.1787	0.1273	0.1280	0.1263
40-50	20.4341	0.1834	0.1270	0.1271	0.1270
30-40	26.1111	0.1888	0.1277	0.1272	0.1275
20-30	31.2596	0.1928	0.1260	0.1254	0.1257
10-20	34.4093	0.1941	0.1269	0.1256	0.1262
0-10	35.5075	0.1933	0.1264	0.1259	0.1262

Figura 7.12 - Tempo de latência máximo para 90% dos agentes, em segundos.

Tabela 7.10 - Número médio de saltos dos agentes dos diferentes algoritmos.

Intervalo	AN	ANFA	TB	sTB	TBu
90-100	45.1013	45.9888	14.2862	14.2107	14.9587
80-90	36.1853	36.8266	15.0425	15.4578	15.4239
70-80	32.7339	33.5342	15.6459	15.6437	15.7261
60-70	27.8138	27.8650	16.1514	16.1234	16.1198
50-60	24.7585	25.0522	16.0338	16.0367	13.9499
40-50	22.4682	24.5953	15.8874	15.8666	15.8631
30-40	16.7735	24.3123	15.9380	15.7309	15.7274
20-30	12.5161	23.6006	15.5062	15.3154	15.3119
10-20	11.4593	23.4201	15.4563	15.2217	15.2183
0-10	14.4811	22.9584	15.0370	14.9014	14.8981

Figura 7.13 - Número médio de saltos dos agentes dos diferentes algoritmos.

Tabela 7.11 - *Largura de banda utilizada pelos agentes em relação à largura de banda total da rede, em 10^{-5} .*

Intervalo	AN	ANFA	TB	sTB	TBu
90-100	10.800	11.017	0.104	0.118	2.495
80-90	8.529	8.747	0.245	0.321	2.665
70-80	7.590	7.838	0.304	0.329	2.776
60-70	6.213	6.278	0.667	0.537	2.912
50-60	4.940	5.562	0.653	0.966	2.856
40-50	3.645	5.443	1.608	1.589	2.721
30-40	2.027	5.365	2.311	2.284	2.682
20-30	0.924	5.205	3.239	3.203	2.602
10-20	0.480	5.136	4.588	4.536	2.568
0-10	0.150	4.956	9.177	9.073	2.478

Figura 7.14 - *Largura de banda utilizada pelos agentes em relação à largura de banda total da rede, em 10^{-5} .*

7.8 Sensibilidade dos Algoritmos em Relação ao Número de Agentes

Os diferentes algoritmos utilizam diferentes métodos para gerar os agentes que analisam a rede. O AN, o ANFA e o TBu utilizam uma abordagem de geração uniforme, enquanto que o TB e o TBs geram seus agentes de acordo com o tráfego atual. Uma maneira de se visualizar os agentes seria como “amostradores” da rede, que amostram as condições da rede e realizam os ajustes necessários de acordo com a condição que amostraram. Seria de se esperar então que com um número maior de agentes, ter-se-ia um melhor desempenho.

Dessa forma, para realizar uma melhor análise dos diferentes algoritmos, faz-se necessário analisar a sensibilidade do desempenho dos algoritmos em relação à taxa de lançamento de agentes para a amostragem da rede. Caso se mantenham todas as condições de tráfego constantes, mudando-se somente a taxa de lançamento de agentes dos algoritmos, obtêm-se os resultados mostrados na figura 7.15, que contém a porcentagem de pacotes entregues em função do número de agentes lançados. É fácil observar que:

1. Aumentando-se o número de agentes, melhora-se o desempenho do algoritmo, até um ponto de saturação, que é dependente da topologia da rede e das características do tráfego na rede.
2. Existe uma separação entre os gráficos de TB, TBs e TBu e AN e ANFA, sendo que estes últimos conseguem atingir o ponto de saturação com um número menor de agentes. Acredita-se que essa diferença seja causada pela diferença de modelos de tráfego utilizados pelos algoritmos. Enquanto o AN e o ANFA utilizam um modelo que armazena a média, variância e melhor tempo para os tempos de viagem dos agentes, o TB, TBs e o TBu possuem um modelo mais simples, que armazena somente a média do valor calculado.

Outro detalhe é a diferença entre a definição dos pontos de operação entre AN, ANFA e TBu e o ponto de operação de TB e TBs. O método de lançamento definido por AN, ANFA e TBu permite a esses algoritmos trabalhar perto do ponto de saturação visto na curva. Isso não ocorre com o TB e o TBs, que nas suas configurações padrões possuem um ponto de operação no meio da curva.

Apesar das diferenças no resultado serem pequenas para o caso da porcentagem de pacotes entregues, a diferença de desempenho causada pela diferença do número de agentes lançados

Figura 7.15 - Porcentagem de pacotes de dados entregue em função do número de agentes lançados para os diferentes algoritmos.

Figura 7.16 - Intervalo máximo de atraso para 90% dos pacotes de dados em função do número de agentes lançados para os diferentes algoritmos.

é crucial para o atraso médio dos pacotes de dados, como pode ser visto na figura 7.16. Como para o caso anterior, novamente verifica-se que o TB e o TBs não encontram o seu ponto de operação perto da saturação. Entretanto, nesse caso, a diferença de agentes que amostram a rede permite que o AN, o ANFA e o TBu tenham um desempenho em muito superior ao do TB e do TBs.

Esse resultado não chega a ser surpreendente, uma vez que o próprio autor do TB escreveu em [13] que não fez um ajuste fino no algoritmo, e que poderia haver melhorias ao se encontrarem parâmetros mais adequados para o funcionamento do algoritmo.

Capítulo 8

Conclusões

Um dos principais requisitos dos atuais sistemas de comunicação e de redes de computadores é suportar diversos tipos de tráfego que possuem características e necessidades de desempenho diferentes, utilizando de forma eficiente os recursos disponíveis, o que justifica a importância do estudo de algoritmos de roteamento. Este trabalho envolveu o estudo dos principais algoritmos de roteamento baseados em formigas e sua comparação com o algoritmo padrão OSPF de roteamento na Internet. Foram propostos dois novos algoritmos visando solucionar alguma das deficiências encontradas.

Para avaliar o desempenho dos diferentes algoritmos foram realizadas extensivas simulações com um simulador de redes desenvolvido em C++. Os experimentos foram comparados utilizando métricas padrões usadas no desempenho de redes, como a porcentagem de pacotes entregues e o atraso dos pacotes, e tendo o OSPF como algoritmo base para comparação. Com base nos resultados obtidos e apresentados no capítulo 7, chega-se às seguintes conclusões:

- Os algoritmos baseados em agentes possuem um desempenho superior aos atuais algoritmos de roteamento: eles conseguem manter uma maior taxa de entrega de pacotes e um melhor desempenho quanto ao atraso dos pacotes quando comparados com o OSPF, utilizando-se para isso de um mínimo de overhead;
- O modelo de algoritmos baseados em agentes é simples e elegante, sendo de fácil compreensão, favorecendo a análise e gerenciamento das redes de comunicação;
- Verificou-se que o desempenho dos diferentes algoritmos baseados em agentes está mais fortemente correlacionado com a taxa de lançamento de agentes do que propriamente

com a diferença entre políticas de roteamento de agentes pelos diversos algoritmos. Dessa forma, com a implementação de modelos mais simples e que possuem uma complexidade computacional menor, como o implementado no TBU, é possível obter os mesmos resultados com um custo menor. Com o aumento constante de dimensão que se tem visto nas redes de comunicação, essa característica não deve ser desprezada, uma vez que para o aumento das taxas de comunicação de dados, a complexidade computacional do algoritmo pode limitar o seu funcionamento no futuro;

- Acredita-se também que a diferenciação dos agentes proposta pelo algoritmo TB pode levar a algoritmos mais eficientes e que possuam uma capacidade de auto-gerenciamento maior do que os algoritmos apresentados. Agentes com diferentes funções são uma das maneiras de se implementar redes com diferentes QoS.

Apesar das vantagens apresentadas pelos algoritmos baseados em agentes, eles ainda apresentam um grande problema: eles não são compatíveis com os atuais algoritmos de roteamento encontrados em funcionamento, ou seja, eles não são capazes de interagir lado a lado com o OSPF, por exemplo. Para que os algoritmos baseados em agentes funcionem, é necessário que todos os servidores da rede entendam e trabalhem utilizando a teoria de reforço do modelo de comunicação por estigmergia. Entretanto, analisando-se historicamente o mercado de computação e de redes, verifica-se que dificilmente um modelo conseguiu se implantar obrigando a total substituição da solução anterior, por melhor que fosse. Normalmente, são utilizadas soluções que permitem a utilização do equipamento ou programa antigo lado a lado com as novas soluções, o que desobriga a total substituição do parque anterior de equipamentos. Esse talvez seja o grande problema dos algoritmos baseados em agentes: como realizar a migração do atual parque instalado para um sistema de comunicação baseado em agentes.

Anexo A

Resultados Completos das Simulações

Neste anexo encontram-se os resultados completos das diversas simulações realizadas para verificar o desempenho dos diferentes algoritmos, para as várias condições de tráfego simuladas. Estes resultados encontram-se nas tabelas seguintes.

Um caso especial é o das simulações para o algoritmo TBu. Como esse algoritmo foi simulado posteriormente a obtenção dos dados dos outros algoritmos, conclui-se que certos casos simulados não tinham interesse na obtenção de resultados. Apesar disso, esses resultados encontram-se presentes, mas são apresentados em tons de cinza, na tabela. Eles não foram obtidos por simulação, mas foram estimados com base nos outros resultados.

Tabela A.1 – Resultados experimentais obtidos para o algoritmo OSPF.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ³)	Bits Gerados (10 ⁶)	Mensagens Entregues (10 ³)	Bits Entregues (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem Entregue	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
128	1,25	28,152	5,872	18,840	3,963	920	66,927%	1,654	2,998	5,521	4,775	25,615	26,406%
128	2,50	14,071	2,934	14,023	2,925	3	99,660%	0,927	1,679	3,093	6,157	18,050	18,608%
128	3,75	9,268	1,933	9,268	1,933	0	99,996%	0,063	0,091	0,180	6,130	11,850	12,215%
128	5,00	7,003	1,459	7,002	1,459	0	99,996%	0,048	0,039	0,098	6,156	8,985	9,262%
128	6,25	5,641	1,176	5,640	1,176	0	99,995%	0,046	0,033	0,089	6,195	7,288	7,513%
128	7,50	4,671	0,974	4,670	0,974	0	99,996%	0,044	0,031	0,084	6,161	6,000	6,185%
128	8,75	3,976	0,829	3,976	0,828	0	99,995%	0,043	0,030	0,083	6,131	5,080	5,236%
128	10,00	3,535	0,737	3,535	0,737	0	99,996%	0,044	0,031	0,083	6,192	4,566	4,707%
256	1,25	55,677	11,677	28,820	6,091	2662	51,762%	2,066	3,079	6,038	4,129	39,742	40,968%
256	2,50	28,026	5,877	18,775	3,963	914	66,991%	1,637	2,936	5,424	4,743	25,596	26,386%
256	3,75	18,602	3,903	13,445	2,830	509	72,278%	1,405	2,777	4,987	4,950	18,799	19,380%
256	5,00	14,095	2,957	13,813	2,900	27	98,001%	1,169	2,116	3,899	6,042	17,857	18,408%
256	6,25	11,099	2,328	11,097	2,328	0	99,981%	0,169	0,461	0,764	6,146	14,309	14,751%
256	7,50	9,304	1,952	9,304	1,952	0	99,995%	0,076	0,153	0,274	6,131	11,969	12,338%
256	8,75	8,030	1,684	8,030	1,684	0	99,995%	0,061	0,101	0,191	6,194	10,430	10,752%
256	10,00	6,978	1,462	6,977	1,462	0	99,995%	0,050	0,057	0,123	6,119	8,946	9,222%
512	1,25	111,593	23,471	36,347	7,706	7462	32,571%	3,161	3,409	7,559	3,247	54,268	55,943%
512	2,50	55,696	11,711	29,072	6,165	2638	52,197%	2,143	3,138	6,190	4,148	39,863	41,094%
512	3,75	37,216	7,826	22,766	4,812	1430	61,172%	1,722	2,922	5,491	4,498	30,879	31,832%
512	5,00	27,777	5,842	18,463	3,905	919	66,467%	1,607	2,882	5,324	4,715	25,345	26,127%
512	6,25	22,273	4,683	15,666	3,303	653	70,332%	1,195	2,531	4,459	4,847	21,354	22,013%
512	7,50	18,570	3,904	13,618	2,873	487	73,330%	1,491	2,847	5,164	5,028	19,168	19,759%
512	8,75	15,748	3,312	13,139	2,772	256	83,442%	1,885	3,003	5,758	5,594	18,167	18,728%
512	10,00	13,819	2,906	13,111	2,761	67	94,935%	1,282	2,294	4,241	5,993	17,355	17,891%

Continuação Tabela A.1 – Resultados experimentais obtidos para o algoritmo OSPF.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ³)	Bits Gerados (10 ¹⁰)	Mensagens Entregues (10 ³)	Bits Entregues (10 ¹⁰)	Mensagens Perdidas (10 ³)	Porcentagem Entregue	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
768	1,25	166,028	34,952	38,036	8,069	12696	22,909%	3,776	3,428	8,198	2,768	62,196	64,116%
768	2,50	83,078	17,491	33,537	7,112	4907	40,370%	2,771	3,324	7,060	3,602	48,053	49,536%
768	3,75	55,335	11,648	28,703	6,093	2638	51,869%	2,139	3,137	6,186	4,126	39,598	40,821%
768	5,00	41,470	8,728	24,469	5,186	1682	59,001%	1,809	3,000	5,679	4,367	33,138	34,161%
768	6,25	33,545	7,062	21,156	4,477	1223	63,067%	1,586	2,829	5,235	4,525	28,754	29,642%
768	7,50	27,995	5,892	18,371	3,887	952	65,631%	1,595	2,856	5,279	4,692	25,397	26,181%
768	8,75	23,681	4,984	16,371	3,459	721	69,138%	1,392	2,676	4,844	4,819	22,332	23,021%
768	10,00	20,760	4,369	14,838	3,135	584	71,478%	1,443	2,792	5,045	4,961	20,562	21,196%
1024	1,25	221,958	46,749	38,596	8,191	18201	17,389%	4,118	3,456	8,576	2,471	66,050	68,088%
1024	2,50	111,197	23,423	35,627	7,565	7489	32,042%	3,168	3,425	7,586	3,234	53,683	55,340%
1024	3,75	73,569	15,494	31,236	6,633	4196	42,466%	2,626	3,305	6,889	3,763	45,279	46,676%
1024	5,00	55,361	11,661	28,326	6,014	2679	51,166%	2,211	3,176	6,308	4,136	39,381	40,597%
1024	6,25	44,153	9,296	25,193	5,342	1878	57,057%	1,858	2,995	5,722	4,326	34,346	35,406%
1024	7,50	37,000	7,794	22,411	4,748	1443	60,582%	1,741	2,964	5,565	4,488	30,729	31,678%
1024	8,75	31,771	6,691	19,949	4,223	1169	62,796%	1,660	2,911	5,415	4,564	27,591	28,443%
1024	10,00	27,690	5,833	18,284	3,873	930	66,045%	1,607	2,903	5,352	4,738	25,253	26,033%
2048	1,25	440,533	92,857	43,154	9,158	39554	9,796%	4,173	3,474	8,654	1,981	74,874	77,185%
2048	2,50	221,043	46,590	38,495	8,182	18120	17,416%	4,119	3,481	8,610	2,503	65,884	67,917%
2048	3,75	146,419	30,868	36,822	7,833	10869	25,149%	3,578	3,488	8,077	2,922	59,192	61,019%
2048	5,00	110,350	23,261	35,770	7,600	7392	32,415%	3,130	3,440	7,568	3,245	53,725	55,383%
2048	6,25	88,633	18,685	32,935	7,003	5520	37,159%	2,787	3,382	7,150	3,480	48,772	50,277%
2048	7,50	73,947	15,584	30,769	6,541	4277	41,611%	2,604	3,348	6,923	3,723	44,880	46,265%
2048	8,75	63,932	13,474	29,527	6,278	3409	46,189%	2,314	3,226	6,475	3,910	41,815	43,106%
2048	10,00	54,894	11,571	27,788	5,906	2688	50,636%	2,101	3,156	6,172	4,098	38,835	40,034%

Tabela A.2 – Resultados experimentais obtidos para o algoritmo AN.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ³)	Bits Gerados (10 ⁶)	Mensagens Entrregues (10 ³)	Bits Entrregues (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
128	1,25	27,800	5,810	27,800	5,810	0	99,966%	0,463	0,629	1,274	8,726	51,400	52,881%
128	2,50	14,000	2,930	14,000	2,930	0	99,977%	0,080	0,092	0,199	8,400	25,500	26,235%
128	3,75	9,314	1,940	9,308	1,940	0	99,939%	0,063	0,063	0,144	8,346	17,300	17,798%
128	5,00	7,111	1,480	7,109	1,480	0	99,972%	0,058	0,052	0,126	8,337	13,500	13,889%
128	6,25	5,594	1,170	5,591	1,170	0	99,946%	0,057	0,050	0,121	8,376	11,000	11,317%
128	7,50	4,656	0,971	4,654	0,970	0	99,954%	0,056	0,058	0,130	8,308	9,270	9,537%
128	8,75	4,040	0,843	4,038	0,843	0	99,952%	0,057	0,051	0,123	8,495	8,390	8,632%
128	10,00	3,434	0,718	3,433	0,717	0	99,978%	0,056	0,054	0,127	8,550	7,350	7,562%
256	1,25	55,600	11,700	35,400	7,420	2000	63,558%	3,108	2,977	6,949	6,129	70,100	72,119%
256	2,50	28,000	5,870	28,000	5,860	1	99,899%	0,645	0,939	1,856	8,821	52,400	53,909%
256	3,75	18,400	3,850	18,400	3,850	0	99,977%	0,113	0,155	0,313	8,468	33,400	34,362%
256	5,00	13,900	2,920	13,900	2,920	1	99,943%	0,079	0,097	0,205	8,394	25,500	26,235%
256	6,25	11,200	2,360	11,200	2,360	0	99,979%	0,066	0,070	0,156	8,213	20,300	20,885%
256	7,50	9,272	1,940	9,270	1,940	0	99,976%	0,061	0,052	0,129	8,293	17,200	17,695%
256	8,75	7,836	1,640	7,832	1,640	0	99,952%	0,061	0,064	0,143	8,393	14,900	15,329%
256	10,00	6,810	1,430	6,808	1,430	0	99,964%	0,058	0,054	0,127	8,301	13,000	13,374%
512	1,25	110,000	23,200	36,500	7,690	7310	32,997%	4,044	3,139	8,093	3,960	79,500	81,790%
512	2,50	55,100	11,600	34,800	7,310	2000	63,090%	3,099	2,976	6,938	6,145	69,700	71,708%
512	3,75	36,900	7,750	31,400	6,610	529	85,263%	2,859	2,977	6,699	7,981	62,700	64,506%
512	5,00	27,800	5,840	27,700	5,830	1	99,831%	0,637	1,013	1,943	8,679	51,400	52,881%
512	6,25	22,200	4,680	22,200	4,670	0	99,956%	0,209	0,359	0,673	8,559	40,800	41,975%
512	7,50	18,600	3,910	18,600	3,910	0	99,979%	0,128	0,179	0,359	8,606	34,400	35,391%
512	8,75	16,100	3,380	16,100	3,380	0	99,964%	0,094	0,138	0,272	8,450	29,500	30,350%
512	10,00	13,900	2,920	13,900	2,920	0	99,984%	0,080	0,094	0,201	8,291	25,100	25,823%

Continuação Tabela A.2 – Resultados experimentais obtidos para o algoritmo AN.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ³)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos Agentes	Overhead de Controle
128	1,25	2,583	62,100	1,418	1,164	28,273	0,400	1,123	6,389
128	2,50	2,583	88,200	0,999	1,584	38,248	0,104	0,209	9,074
128	3,75	2,584	102,000	0,851	1,733	43,863	0,097	0,184	10,494
128	5,00	2,584	110,000	0,778	1,806	47,132	0,095	0,177	11,317
128	6,25	2,584	114,000	0,742	1,841	48,673	0,094	0,174	11,728
128	7,50	2,584	117,000	0,712	1,872	50,177	0,094	0,172	12,037
128	8,75	2,584	119,000	0,694	1,890	50,985	0,094	0,173	12,243
128	10,00	2,584	121,000	0,674	1,909	51,980	0,094	0,171	12,449
256	1,25	2,584	46,700	1,401	1,166	24,488	6,511	15,716	4,805
256	2,50	2,584	60,800	1,425	1,157	27,820	0,535	1,607	6,255
256	3,75	2,583	77,600	1,149	1,434	34,116	0,122	0,272	7,984
256	5,00	2,584	88,300	1,003	1,581	38,242	0,103	0,205	9,084
256	6,25	2,584	96,400	0,910	1,674	41,552	0,096	0,184	9,918
256	7,50	2,583	103,000	0,849	1,734	43,941	0,097	0,183	10,597
256	8,75	2,584	107,000	0,807	1,776	45,781	0,096	0,181	11,008
256	10,00	2,584	110,000	0,774	1,810	47,260	0,095	0,176	11,317
512	1,25	2,584	16,400	0,884	1,670	14,860	13,263	27,318	1,687
512	2,50	2,584	46,200	1,382	1,187	24,343	6,431	15,660	4,753
512	3,75	2,584	55,700	1,461	1,114	26,341	3,425	9,145	5,730
512	5,00	2,584	62,200	1,410	1,172	28,440	0,521	1,667	6,399
512	6,25	2,584	69,500	1,267	1,316	31,071	0,189	0,553	7,150
512	7,50	2,584	76,100	1,162	1,422	33,688	0,129	0,301	7,829
512	8,75	2,584	82,900	1,071	1,513	36,185	0,112	0,245	8,529
512	10,00	2,584	88,800	0,994	1,590	38,484	0,103	0,209	9,136

Continuação Tabela A.2 – Resultados experimentais obtidos para o algoritmo AN.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ³)	Bits Gerados (10 ⁶)	Mensagens Entrinques (10 ³)	Bits Entrinques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
768	1,25	167,000	35,100	30,900	6,540	13400	18,544%	4,939	3,135	8,982	2,788	84,500	86,934%
768	2,50	83,700	17,600	36,100	7,630	4690	43,175%	3,657	3,196	7,780	4,619	74,600	76,749%
768	3,75	55,200	11,600	35,400	7,450	1960	64,031%	3,071	2,975	6,909	6,094	69,900	71,914%
768	5,00	41,500	8,750	32,300	6,810	903	77,813%	2,876	3,038	6,794	7,305	64,700	66,564%
768	6,25	33,100	6,980	30,600	6,440	245	92,321%	2,453	2,765	6,019	8,522	60,500	62,243%
768	7,50	27,900	5,880	27,700	5,840	16	99,332%	0,990	1,595	3,047	8,843	52,700	54,218%
768	8,75	23,500	4,950	23,500	4,950	0	99,976%	0,265	0,426	0,815	8,678	43,600	44,856%
768	10,00	20,900	4,400	20,900	4,400	0	99,966%	0,192	0,370	0,670	8,544	38,300	39,403%
1024	1,25	223,000	47,000	27,900	5,910	19400	12,477%	5,240	3,125	9,271	2,242	88,500	91,049%
1024	2,50	111,000	23,500	36,200	7,660	7440	32,530%	3,964	3,171	8,054	3,916	79,000	81,276%
1024	3,75	73,500	15,500	35,400	7,460	3770	48,088%	3,527	3,133	7,568	5,128	72,500	74,588%
1024	5,00	55,400	11,700	34,100	7,190	2100	61,550%	3,146	3,031	7,055	6,090	69,200	71,193%
1024	6,25	44,700	9,420	33,000	6,970	1150	73,887%	2,850	3,021	6,747	6,958	65,700	67,593%
1024	7,50	37,700	7,930	31,300	6,600	621	83,105%	2,886	3,049	6,819	7,798	62,700	64,506%
1024	8,75	31,500	6,640	29,400	6,190	207	93,200%	2,169	2,624	5,555	8,604	57,900	59,568%
1024	10,00	27,800	5,850	27,700	5,840	6	99,756%	0,849	1,345	2,584	8,816	52,300	53,807%
2048	1,25	439,000	92,500	21,600	4,590	41400	4,935%	5,951	2,605	9,311	1,468	94,100	96,811%
2048	2,50	220,000	46,500	29,300	6,220	18900	13,307%	5,139	3,134	9,182	2,373	87,100	89,609%
2048	3,75	146,000	30,700	35,900	7,590	10900	24,617%	4,440	3,183	8,546	3,282	82,400	84,774%
2048	5,00	110,000	23,200	36,200	7,650	7310	32,861%	4,062	3,208	8,200	3,951	78,800	81,070%
2048	6,25	86,700	18,300	35,300	7,460	5080	40,675%	3,736	3,204	7,870	4,590	74,600	76,749%
2048	7,50	75,000	15,800	35,300	7,470	3910	47,109%	3,508	3,138	7,557	5,064	72,500	74,588%
2048	8,75	64,100	13,500	34,400	7,270	2930	53,642%	3,373	3,100	7,372	5,562	71,300	73,354%
2048	10,00	53,900	11,400	33,800	7,140	1970	62,734%	3,146	3,043	7,072	6,291	68,600	70,576%

Continuação Tabela A.2 – Resultados experimentais obtidos para o algoritmo AN.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ³)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% Agentes	Overhead de Controle
768	1,25	2,584	7,760	0,550	2,001	12,061	17,687	32,786	0,798
768	2,50	2,584	30,900	1,096	1,464	21,359	10,084	22,621	3,179
768	3,75	2,583	46,500	1,382	1,186	24,536	6,226	15,203	4,784
768	5,00	2,584	53,200	1,450	1,122	25,755	4,181	10,945	5,473
768	6,25	2,584	57,300	1,470	1,108	26,611	2,522	7,087	5,895
768	7,50	2,584	61,500	1,422	1,160	28,118	0,811	2,737	6,327
768	8,75	2,584	67,600	1,314	1,269	30,409	0,224	0,653	6,955
768	10,00	2,584	71,900	1,232	1,351	32,060	0,167	0,481	7,397
1024	1,25	2,584	4,616	0,371	2,182	11,367	19,374	34,962	0,475
1024	2,50	2,584	16,000	0,860	1,696	14,919	13,085	27,238	1,646
1024	3,75	2,583	34,600	1,173	1,390	22,264	8,702	20,174	3,560
1024	5,00	2,584	46,500	1,367	1,201	24,618	6,503	15,871	4,784
1024	6,25	2,584	51,600	1,432	1,140	25,426	4,546	11,734	5,309
1024	7,50	2,584	55,300	1,452	1,123	26,264	3,590	9,539	5,689
1024	8,75	2,584	59,800	1,447	1,132	27,653	2,129	6,418	6,152
1024	10,00	2,584	61,600	1,408	1,175	28,211	0,708	2,320	6,337
2048	1,25	2,584	1,460	0,135	2,429	14,481	18,306	35,508	0,150
2048	2,50	2,584	4,710	0,375	2,178	11,552	18,227	33,857	0,485
2048	3,75	2,584	10,200	0,654	1,900	12,971	14,916	29,733	1,049
2048	5,00	2,583	17,300	0,862	1,692	15,981	12,917	27,143	1,780
2048	6,25	2,584	29,100	1,043	1,520	21,334	10,084	22,746	2,994
2048	7,50	2,583	35,100	1,171	1,390	22,613	8,788	20,391	3,611
2048	8,75	2,584	41,100	1,286	1,276	23,637	7,854	18,550	4,228
2048	10,00	2,583	45,400	1,369	1,197	24,143	6,030	14,909	4,671

Tabela A.3 – Resultados experimentais obtidos para o algoritmo ANFA.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
128	1,25	28,143	5,869	28,117	5,864	0	99,908%	0,476	0,621	1,277	8,744	51,994	53,491%
128	2,50	13,973	2,915	13,967	2,914	1	99,953%	0,083	0,115	0,232	8,172	24,827	25,542%
128	3,75	9,451	1,971	9,442	1,969	1	99,903%	0,062	0,058	0,136	8,048	17,068	17,559%
128	5,00	6,963	1,451	6,960	1,450	0	99,952%	0,059	0,081	0,163	8,102	12,914	13,286%
128	6,25	5,500	1,147	5,498	1,147	0	99,964%	0,058	0,067	0,145	8,324	10,727	11,036%
128	7,50	4,696	0,979	4,694	0,978	0	99,943%	0,054	0,047	0,114	8,043	9,117	9,380%
128	8,75	4,010	0,837	4,006	0,836	0	99,913%	0,054	0,048	0,116	8,126	8,061	8,293%
128	10,00	3,521	0,735	3,519	0,735	0	99,944%	0,055	0,048	0,116	8,236	7,311	7,521%
256	1,25	55,977	11,737	35,071	7,381	2060	62,653%	2,655	2,982	6,502	6,142	70,990	73,035%
256	2,50	27,977	5,867	27,957	5,863	1	99,928%	0,528	0,731	1,471	8,699	51,759	53,250%
256	3,75	18,568	3,896	18,564	3,895	0	99,977%	0,131	0,222	0,417	8,318	33,235	34,193%
256	5,00	13,849	2,904	13,843	2,903	0	99,957%	0,080	0,106	0,217	8,044	24,370	25,072%
256	6,25	11,070	2,320	11,068	2,319	0	99,981%	0,068	0,080	0,172	8,058	19,735	20,303%
256	7,50	9,435	1,978	9,433	1,977	0	99,982%	0,061	0,060	0,137	7,863	16,605	17,084%
256	8,75	7,834	1,644	7,831	1,644	0	99,960%	0,059	0,083	0,166	8,104	14,463	14,880%
256	10,00	6,927	1,453	6,923	1,453	0	99,952%	0,057	0,050	0,121	8,082	12,927	13,300%
512	1,25	111,035	23,350	38,242	8,075	7192	34,442%	4,030	3,213	8,175	3,828	80,224	82,535%
512	2,50	55,699	11,714	34,424	7,265	2100	61,805%	2,619	2,981	6,464	6,030	69,955	71,970%
512	3,75	36,744	7,725	28,654	6,033	791	77,982%	3,068	3,464	7,537	7,380	60,296	62,033%
512	5,00	27,612	5,806	27,575	5,799	1	99,867%	0,791	1,158	2,285	8,996	52,956	54,481%
512	6,25	22,051	4,639	22,040	4,636	0	99,952%	0,211	0,335	0,643	8,515	40,289	41,450%
512	7,50	18,574	3,906	18,571	3,905	0	99,983%	0,123	0,182	0,358	8,350	33,425	34,388%
512	8,75	16,124	3,391	16,118	3,389	0	99,963%	0,113	0,223	0,401	8,247	28,908	29,741%
512	10,00	13,688	2,877	13,686	2,877	0	99,985%	0,076	0,080	0,179	8,036	24,066	24,759%

Continuação Tabela A.3 – Resultados experimentais obtidos para o algoritmo ANFA.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
128	1,25	2,583	61,892	1,375	1,209	28,138	0,088	0,174	6,367
128	2,50	2,583	91,165	0,958	1,625	39,406	0,091	0,176	9,379
128	3,75	2,584	104,121	0,820	1,763	44,601	0,094	0,179	10,712
128	5,00	2,584	111,689	0,752	1,831	47,779	0,093	0,173	11,491
128	6,25	2,584	115,451	0,713	1,870	49,490	0,094	0,173	11,878
128	7,50	2,583	118,879	0,687	1,896	51,078	0,094	0,172	12,230
128	8,75	2,584	119,919	0,678	1,905	51,586	0,093	0,170	12,337
128	10,00	2,584	121,686	0,666	1,918	52,446	0,092	0,167	12,519
256	1,25	2,584	53,365	1,491	1,092	24,796	0,090	0,180	5,490
256	2,50	2,584	62,472	1,365	1,219	28,443	0,088	0,174	6,427
256	3,75	2,584	79,697	1,104	1,479	34,905	0,090	0,174	8,199
256	5,00	2,584	92,673	0,942	1,642	39,989	0,092	0,176	9,534
256	6,25	2,583	99,723	0,866	1,717	42,810	0,092	0,176	10,260
256	7,50	2,584	104,060	0,818	1,765	44,531	0,093	0,177	10,706
256	8,75	2,584	109,127	0,774	1,809	46,691	0,095	0,177	11,227
256	10,00	2,584	111,685	0,757	1,827	47,745	0,094	0,174	11,490
512	1,25	2,584	51,754	1,560	1,023	24,173	0,095	0,190	5,324
512	2,50	2,583	53,158	1,486	1,098	24,688	0,090	0,180	5,469
512	3,75	2,584	58,102	1,381	1,203	26,744	0,089	0,176	5,978
512	5,00	2,584	62,075	1,368	1,216	28,193	0,089	0,176	6,386
512	6,25	2,584	72,354	1,220	1,364	32,042	0,089	0,174	7,444
512	7,50	2,584	79,325	1,108	1,476	34,806	0,090	0,175	8,161
512	8,75	2,584	85,024	1,035	1,549	36,827	0,091	0,178	8,747
512	10,00	2,584	91,966	0,945	1,639	39,703	0,091	0,175	9,462

Continuação Tabela A.3 – Resultados experimentais obtidos para o algoritmo ANFA.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
768	1,25	166,934	35,140	35,396	7,494	13027	21,204%	4,807	3,237	8,982	2,897	84,268	86,696%
768	2,50	83,218	17,517	36,392	7,703	4635	43,730%	3,483	3,289	7,726	4,584	76,236	78,432%
768	3,75	55,269	11,636	34,292	7,243	2066	62,046%	2,719	2,983	6,567	6,121	70,239	72,262%
768	5,00	42,416	8,928	29,079	6,137	1313	68,556%	2,414	3,296	6,666	6,594	61,402	63,170%
768	6,25	33,353	7,022	29,609	6,237	364	88,775%	2,905	3,213	7,050	8,223	60,180	61,914%
768	7,50	28,088	5,912	27,909	5,874	16	99,363%	1,072	1,616	3,157	8,981	53,978	55,533%
768	8,75	24,020	5,055	24,012	5,054	0	99,965%	0,276	0,461	0,872	8,603	44,217	45,491%
768	10,00	20,346	4,280	20,338	4,278	0	99,965%	0,163	0,258	0,496	8,534	37,340	38,415%
1024	1,25	221,887	46,735	32,626	6,911	18761	14,704%	4,972	3,229	9,138	2,409	86,482	88,973%
1024	2,50	109,975	23,157	37,989	8,033	7114	34,544%	4,089	3,183	8,194	3,904	80,044	82,349%
1024	3,75	73,579	15,497	35,374	7,480	3774	48,076%	3,549	3,346	7,865	4,920	73,861	75,989%
1024	5,00	55,874	11,763	34,459	7,285	2115	61,672%	2,735	3,025	6,637	5,978	69,807	71,818%
1024	6,25	43,776	9,220	29,991	6,335	1362	68,510%	2,492	3,266	6,705	6,639	63,365	65,190%
1024	7,50	36,831	7,755	28,549	6,019	813	77,512%	2,873	3,283	7,109	7,406	60,399	62,139%
1024	8,75	31,623	6,659	29,803	6,275	177	94,245%	2,378	2,629	5,769	8,765	59,972	61,700%
1024	10,00	27,291	5,752	27,244	5,742	4	99,826%	0,884	1,504	2,824	8,940	52,230	53,734%
2048	1,25	437,487	92,216	26,855	5,688	40824	6,138%	5,632	2,839	9,294	1,665	92,139	94,793%
2048	2,50	219,450	46,259	32,413	6,866	18534	14,770%	4,957	3,268	9,173	2,460	86,307	88,794%
2048	3,75	147,046	30,993	35,930	7,620	10998	24,435%	4,571	3,261	8,778	3,183	82,569	84,948%
2048	5,00	110,457	23,283	35,671	7,559	7395	32,294%	4,132	3,290	8,376	3,893	79,542	81,833%
2048	6,25	88,582	18,672	36,476	7,728	5158	41,177%	3,696	3,321	7,981	4,397	76,254	78,451%
2048	7,50	72,250	15,229	34,224	7,250	3758	47,369%	3,446	3,315	7,722	5,187	73,737	75,862%
2048	8,75	62,535	13,181	33,582	7,109	2861	53,701%	3,435	3,263	7,644	5,653	71,806	73,874%
2048	10,00	54,657	11,518	34,401	7,276	2000	62,939%	2,973	3,151	7,037	5,969	69,207	71,200%

Continuação Tabela A.3 – Resultados experimentais obtidos para o algoritmo ANFA.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
768	1,25	2,584	50,161	1,610	0,974	23,426	0,095	0,193	5,161
768	2,50	2,584	52,791	1,523	1,061	24,546	0,093	0,186	5,431
768	3,75	2,583	53,341	1,478	1,105	24,675	0,090	0,179	5,488
768	5,00	2,583	56,724	1,377	1,207	26,275	0,089	0,177	5,836
768	6,25	2,584	58,470	1,387	1,197	26,997	0,089	0,176	6,015
768	7,50	2,584	60,971	1,372	1,211	27,829	0,088	0,175	6,273
768	8,75	2,584	67,853	1,271	1,312	30,494	0,089	0,175	6,981
768	10,00	2,584	73,362	1,185	1,399	32,383	0,089	0,174	7,548
1024	1,25	2,583	50,068	1,617	0,966	23,464	0,096	0,195	5,150
1024	2,50	2,584	51,505	1,562	1,022	24,033	0,094	0,189	5,299
1024	3,75	2,584	53,091	1,504	1,080	24,716	0,092	0,184	5,462
1024	5,00	2,584	52,760	1,489	1,095	24,445	0,090	0,179	5,428
1024	6,25	2,584	55,917	1,416	1,168	25,833	0,089	0,177	5,753
1024	7,50	2,584	58,122	1,378	1,205	26,852	0,089	0,176	5,980
1024	8,75	2,584	57,876	1,412	1,172	26,580	0,089	0,177	5,954
1024	10,00	2,583	62,409	1,362	1,221	28,381	0,089	0,174	6,421
2048	1,25	2,584	48,174	1,681	0,903	22,958	0,096	0,193	4,956
2048	2,50	2,584	49,792	1,621	0,962	23,376	0,096	0,194	5,123
2048	3,75	2,584	51,021	1,584	0,999	23,775	0,095	0,192	5,249
2048	5,00	2,584	52,338	1,559	1,025	24,347	0,095	0,191	5,385
2048	6,25	2,584	52,987	1,531	1,053	24,696	0,093	0,185	5,451
2048	7,50	2,584	53,207	1,501	1,083	24,769	0,092	0,184	5,474
2048	8,75	2,584	52,535	1,501	1,082	24,350	0,090	0,180	5,405
2048	10,00	2,583	53,195	1,485	1,098	24,654	0,090	0,180	5,473

Tabela A.4 – Resultados experimentais obtidos para o algoritmo TB.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
128	1,25	28,073	5,856	26,918	5,615	111	95,888%	0,935	1,588	2,984	7,113	41,098	42,281%
128	2,50	14,123	2,945	14,122	2,945	0	99,996%	0,179	0,442	0,749	6,539	19,292	19,847%
128	3,75	9,348	1,950	9,347	1,950	0	99,989%	0,057	0,066	0,142	6,258	12,222	12,574%
128	5,00	7,073	1,476	7,073	1,476	0	99,999%	0,047	0,035	0,092	6,266	9,262	9,529%
128	6,25	5,543	1,156	5,543	1,156	0	99,995%	0,046	0,032	0,088	6,294	7,291	7,501%
128	7,50	4,651	0,970	4,651	0,970	0	99,997%	0,045	0,031	0,085	6,275	6,092	6,267%
128	8,75	3,953	0,825	3,953	0,825	0	99,999%	0,043	0,030	0,083	6,160	5,083	5,229%
128	10,00	3,410	0,712	3,409	0,712	0	99,994%	0,044	0,030	0,083	6,238	4,444	4,572%
256	1,25	55,565	11,654	36,122	7,595	1910	65,008%	2,652	2,971	6,485	5,321	55,951	57,563%
256	2,50	27,996	5,871	26,205	5,499	177	93,601%	1,122	1,894	3,565	7,183	41,135	42,320%
256	3,75	18,435	3,869	18,041	3,786	39	97,863%	0,613	1,582	2,654	7,002	26,970	27,747%
256	5,00	13,926	2,920	13,925	2,920	0	99,992%	0,230	0,645	1,063	6,700	19,589	20,153%
256	6,25	11,079	2,324	11,077	2,323	0	99,982%	0,111	0,250	0,433	6,521	15,163	15,599%
256	7,50	9,189	1,927	9,188	1,927	0	99,992%	0,062	0,087	0,174	6,453	12,450	12,808%
256	8,75	7,920	1,661	7,919	1,661	0	99,994%	0,054	0,066	0,139	6,345	10,549	10,853%
256	10,00	6,935	1,453	6,935	1,453	0	99,995%	0,048	0,035	0,093	6,305	9,173	9,437%
512	1,25	110,810	23,305	40,228	8,504	6982	36,303%	3,491	3,238	7,668	3,847	68,149	70,112%
512	2,50	56,145	11,803	35,190	7,411	2062	62,678%	2,775	3,020	6,671	5,272	55,915	57,526%
512	3,75	36,909	7,768	30,190	6,361	656	81,796%	2,341	2,941	6,134	6,360	48,356	49,748%
512	5,00	27,723	5,829	26,097	5,488	157	94,136%	1,152	2,026	3,766	7,038	40,325	41,487%
512	6,25	22,415	4,714	21,555	4,533	86	96,163%	0,557	1,460	2,440	6,972	32,715	33,657%
512	7,50	18,558	3,900	18,162	3,818	38	97,867%	0,626	1,720	2,845	6,869	26,785	27,557%
512	8,75	16,055	3,376	15,903	3,344	15	99,051%	0,288	0,982	1,554	6,723	22,723	23,377%
512	10,00	14,029	2,952	13,962	2,939	6	99,523%	0,556	1,464	2,445	6,969	20,628	21,222%

Continuação Tabela A.4 – Resultados experimentais obtidos para o algoritmo TB.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
128	1,25	0,557	5,987	0,525	0,031	16,011	0,068	0,125	0,616
128	2,50	0,272	2,467	0,258	0,014	14,661	0,062	0,117	0,254
128	3,75	0,176	1,462	0,167	0,009	13,993	0,061	0,116	0,150
128	5,00	0,130	1,079	0,124	0,006	14,029	0,060	0,116	0,111
128	6,25	0,101	0,832	0,095	0,005	14,073	0,061	0,117	0,086
128	7,50	0,083	0,669	0,079	0,004	13,875	0,061	0,116	0,069
128	8,75	0,070	0,554	0,067	0,003	13,736	0,059	0,113	0,057
128	10,00	0,060	0,475	0,057	0,003	13,749	0,060	0,114	0,049
256	1,25	1,115	0,002	1,049	0,066	15,703	0,068	0,125	0,000
256	2,50	0,556	6,180	0,523	0,032	16,284	0,069	0,129	0,636
256	3,75	0,361	3,572	0,343	0,018	15,655	0,065	0,123	0,367
256	5,00	0,270	0,835	0,256	0,014	14,966	0,063	0,119	0,086
256	6,25	0,212	1,837	0,202	0,011	14,561	0,062	0,118	0,189
256	7,50	0,175	1,521	0,165	0,009	14,409	0,062	0,119	0,156
256	8,75	0,150	0,000	0,142	0,007	14,193	0,062	0,119	0,000
256	10,00	0,131	0,000	0,124	0,007	14,081	0,061	0,118	0,000
512	1,25	2,234	23,093	2,100	0,134	15,753	0,069	0,126	2,376
512	2,50	1,126	11,901	1,059	0,066	15,926	0,069	0,127	1,224
512	3,75	0,735	8,169	0,692	0,043	16,252	0,069	0,129	0,840
512	5,00	0,549	4,852	0,518	0,030	16,045	0,068	0,127	0,499
512	6,25	0,442	4,649	0,418	0,024	15,807	0,067	0,125	0,478
512	7,50	0,365	3,608	0,346	0,019	15,278	0,064	0,121	0,371
512	8,75	0,313	2,383	0,297	0,016	15,043	0,063	0,118	0,245
512	10,00	0,273	1,361	0,258	0,015	15,396	0,065	0,124	0,140

Continuação Tabela A.4 – Resultados experimentais obtidos para o algoritmo TB.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ¹⁰)	Mensagens Entrques (10 ⁵)	Bits Entrques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
768	1,25	164,386	34,606	40,141	8,499	12304	24,419%	4,170	3,302	8,430	3,139	75,155	77,320%
768	2,50	83,397	17,558	38,106	8,053	4474	45,693%	3,167	3,247	7,356	4,280	62,408	64,206%
768	3,75	55,505	11,682	34,697	7,320	2049	62,511%	2,771	3,053	6,709	5,271	56,058	57,672%
768	5,00	40,753	8,582	30,814	6,497	976	75,612%	2,428	3,079	6,400	6,028	49,621	51,050%
768	6,25	33,914	7,137	30,295	6,378	354	89,328%	2,263	2,739	5,796	6,740	47,399	48,764%
768	7,50	27,344	5,754	26,308	5,537	101	96,211%	1,294	2,021	3,902	7,310	41,769	42,973%
768	8,75	23,647	4,979	22,843	4,812	80	96,599%	0,625	1,513	2,577	7,061	34,800	35,802%
768	10,00	20,871	4,393	20,449	4,305	42	97,979%	0,669	1,664	2,815	7,094	31,105	32,001%
1024	1,25	221,062	46,565	38,375	8,138	18120	17,359%	4,508	3,321	8,792	2,676	79,036	81,312%
1024	2,50	110,959	23,371	39,015	8,262	7126	35,162%	3,636	3,319	7,918	3,787	68,035	69,995%
1024	3,75	73,991	15,588	35,995	7,613	3748	48,648%	3,237	3,275	7,462	4,560	60,562	62,306%
1024	5,00	55,259	11,644	34,435	7,272	2047	62,317%	2,840	3,087	6,823	5,376	56,313	57,935%
1024	6,25	45,086	9,493	31,725	6,692	1316	70,364%	2,610	3,134	6,652	5,730	52,343	53,851%
1024	7,50	37,153	7,825	29,636	6,249	737	79,767%	2,497	3,022	6,395	6,300	47,469	48,837%
1024	8,75	31,615	6,656	28,163	5,934	337	89,082%	2,258	2,785	5,850	6,918	45,302	46,607%
1024	10,00	27,643	5,821	26,246	5,530	135	94,948%	1,437	2,256	4,348	7,195	41,453	42,647%
2048	1,25	437,333	92,185	36,120	7,661	39907	8,259%	5,191	3,097	9,185	1,931	87,798	90,327%
2048	2,50	220,496	46,474	38,473	8,164	18062	17,448%	4,493	3,342	8,805	2,697	78,681	80,948%
2048	3,75	144,359	30,426	40,012	8,484	10336	27,717%	3,664	3,369	8,010	3,309	71,927	73,999%
2048	5,00	110,958	23,394	37,852	8,022	7245	34,114%	3,559	3,345	7,874	3,796	66,916	68,843%
2048	6,25	89,651	18,898	37,213	7,883	5186	41,509%	3,406	3,328	7,699	4,173	63,880	65,720%
2048	7,50	73,633	15,519	36,972	7,824	3624	50,211%	3,106	3,208	7,245	4,584	60,816	62,568%
2048	8,75	63,147	13,311	36,464	7,707	2628	57,746%	2,862	3,165	6,945	4,867	58,245	59,923%
2048	10,00	54,898	11,564	34,554	7,302	2006	62,942%	2,827	3,081	6,801	5,233	55,419	57,015%

Continuação Tabela A.4 – Resultados experimentais obtidos para o algoritmo TB.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
768	1,25	3,311	33,145	3,109	0,202	15,420	0,069	0,126	3,410
768	2,50	1,676	17,224	1,575	0,101	15,637	0,068	0,125	1,772
768	3,75	1,111	12,096	1,044	0,067	16,141	0,070	0,128	1,244
768	5,00	0,813	8,765	0,765	0,048	16,034	0,069	0,127	0,902
768	6,25	0,676	7,283	0,637	0,038	15,944	0,068	0,127	0,749
768	7,50	0,544	6,147	0,514	0,030	16,399	0,070	0,131	0,632
768	8,75	0,470	5,042	0,445	0,026	15,908	0,067	0,126	0,519
768	10,00	0,414	0,000	0,392	0,022	15,844	0,066	0,124	0,000
1024	1,25	4,447	44,867	4,176	0,271	15,458	0,070	0,127	4,616
1024	2,50	2,227	23,436	2,091	0,136	15,875	0,070	0,128	2,411
1024	3,75	1,482	16,086	1,392	0,090	16,094	0,069	0,128	1,655
1024	5,00	1,102	0,000	1,037	0,065	16,206	0,070	0,128	0,000
1024	6,25	0,900	0,090	0,845	0,054	16,318	0,070	0,130	0,009
1024	7,50	0,741	8,050	0,697	0,043	16,109	0,069	0,127	0,828
1024	8,75	0,628	7,084	0,593	0,035	16,383	0,069	0,129	0,729
1024	10,00	0,549	6,076	0,517	0,031	16,178	0,069	0,130	0,625
2048	1,25	8,773	0,012	8,248	0,524	15,037	0,070	0,126	0,001
2048	2,50	4,418	44,329	4,149	0,269	15,455	0,070	0,126	4,561
2048	3,75	2,887	29,830	2,713	0,174	15,592	0,069	0,126	3,069
2048	5,00	2,218	23,549	2,080	0,137	15,927	0,070	0,127	2,423
2048	6,25	1,791	19,756	1,684	0,106	16,197	0,071	0,130	2,033
2048	7,50	1,470	15,721	1,380	0,090	15,960	0,070	0,128	1,617
2048	8,75	1,260	13,473	1,184	0,076	15,859	0,069	0,128	1,386
2048	10,00	1,094	11,581	1,030	0,064	15,908	0,069	0,127	1,191

Tabela A.5 – Resultados experimentais obtidos para o algoritmo TBs.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
128	1,25	27,956	5,834	25,786	5,383	214	92,237%	0,823	1,706	3,024	7,089	40,358	41,520%
128	2,50	14,111	2,945	14,109	2,944	0	99,987%	0,213	0,587	0,971	6,906	20,363	20,949%
128	3,75	9,228	1,925	9,227	1,925	0	99,994%	0,055	0,055	0,125	6,482	12,491	12,850%
128	5,00	6,945	1,448	6,945	1,448	0	99,996%	0,048	0,035	0,093	6,355	9,207	9,472%
128	6,25	5,604	1,168	5,604	1,168	0	99,997%	0,045	0,032	0,087	6,323	7,397	7,610%
128	7,50	4,604	0,960	4,604	0,960	0	99,997%	0,045	0,033	0,087	6,334	6,084	6,259%
128	8,75	3,997	0,834	3,996	0,834	0	99,997%	0,044	0,031	0,084	6,220	5,189	5,339%
128	10,00	3,544	0,739	3,544	0,739	0	99,997%	0,045	0,031	0,085	6,429	4,756	4,893%
256	1,25	55,418	11,626	35,561	7,473	1956	64,169%	2,853	3,017	6,745	5,473	58,185	59,861%
256	2,50	28,010	5,872	26,279	5,511	169	93,820%	1,254	2,000	3,834	7,289	41,867	43,073%
256	3,75	18,600	3,896	18,250	3,822	35	98,115%	0,592	1,637	2,704	6,967	27,138	27,920%
256	5,00	13,913	2,919	13,910	2,918	0	99,976%	0,194	0,487	0,822	6,681	19,519	20,081%
256	6,25	11,259	2,363	11,255	2,362	0	99,969%	0,105	0,218	0,386	6,695	15,829	16,285%
256	7,50	9,426	1,978	9,424	1,978	0	99,987%	0,074	0,201	0,333	6,500	12,871	13,241%
256	8,75	7,889	1,654	7,888	1,654	0	99,995%	0,050	0,038	0,099	6,379	10,566	10,870%
256	10,00	6,943	1,455	6,942	1,455	0	99,997%	0,049	0,038	0,098	6,492	9,461	9,733%
512	1,25	110,348	23,210	38,272	8,093	7132	34,683%	3,584	3,299	7,840	3,839	70,299	72,324%
512	2,50	55,157	11,598	34,472	7,264	2041	62,498%	2,787	3,092	6,776	5,451	57,618	59,277%
512	3,75	37,685	7,923	30,570	6,434	692	81,120%	2,543	3,039	6,464	6,562	50,771	52,234%
512	5,00	27,568	5,800	25,606	5,389	194	92,881%	1,199	2,107	3,916	7,091	40,318	41,480%
512	6,25	21,917	4,607	21,310	4,481	58	97,231%	0,654	1,557	2,663	7,357	33,729	34,701%
512	7,50	18,971	3,988	18,744	3,941	22	98,807%	0,541	1,459	2,423	7,265	28,995	29,830%
512	8,75	15,776	3,318	15,775	3,317	0	99,995%	0,322	0,910	1,495	7,172	23,818	24,504%
512	10,00	14,148	2,977	14,111	2,970	2	99,740%	0,327	0,936	1,534	7,123	21,222	21,834%

Continuação Tabela A.5 – Resultados experimentais obtidos para o algoritmo TBs.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ³)	Agentes Recabidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% Agentes	Overhead de Controle
128	1,25	0,554	5,950	0,523	0,031	15,953	0,067	0,125	0,612
128	2,50	0,273	2,581	0,259	0,013	15,006	0,063	0,118	0,266
128	3,75	0,173	1,461	0,165	0,009	14,095	0,060	0,115	0,150
128	5,00	0,128	1,040	0,121	0,007	13,896	0,060	0,114	0,107
128	6,25	0,101	0,814	0,096	0,005	13,785	0,059	0,115	0,084
128	7,50	0,082	0,657	0,078	0,004	13,808	0,061	0,117	0,068
128	8,75	0,071	0,547	0,067	0,004	13,506	0,059	0,114	0,056
128	10,00	0,062	0,501	0,059	0,003	13,952	0,061	0,116	0,052
256	1,25	1,112	0,000	1,044	0,067	15,895	0,069	0,127	0,000
256	2,50	0,556	5,826	0,525	0,031	16,213	0,069	0,128	0,599
256	3,75	0,365	0,000	0,346	0,019	15,253	0,064	0,121	0,000
256	5,00	0,270	0,000	0,256	0,014	14,501	0,061	0,115	0,000
256	6,25	0,216	1,924	0,205	0,011	14,610	0,062	0,116	0,198
256	7,50	0,179	1,509	0,171	0,009	14,151	0,061	0,116	0,155
256	8,75	0,149	0,000	0,142	0,007	13,881	0,060	0,116	0,000
256	10,00	0,131	1,085	0,124	0,006	14,123	0,061	0,118	0,112
512	1,25	2,224	7,610	2,089	0,135	15,793	0,070	0,128	0,783
512	2,50	1,106	12,038	1,039	0,067	16,191	0,070	0,129	1,239
512	3,75	0,751	5,971	0,707	0,043	16,250	0,070	0,130	0,614
512	5,00	0,546	5,806	0,516	0,030	15,925	0,068	0,127	0,597
512	6,25	0,432	4,676	0,408	0,024	15,969	0,067	0,127	0,481
512	7,50	0,373	3,920	0,354	0,019	15,756	0,066	0,124	0,403
512	8,75	0,308	3,116	0,292	0,016	15,458	0,064	0,122	0,321
512	10,00	0,275	2,759	0,262	0,014	15,426	0,064	0,120	0,284

Continuação Tabela A.5 – Resultados experimentais obtidos para o algoritmo TBs.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ⁵)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
768	1,25	166,341	35,020	37,621	7,972	12756	22,617%	4,426	3,342	8,737	3,023	77,378	79,607%
768	2,50	82,771	17,423	37,646	7,952	4461	45,482%	3,230	3,248	7,420	4,422	64,618	66,479%
768	3,75	54,862	11,553	34,331	7,251	2028	62,577%	2,764	3,104	6,768	5,432	57,437	59,091%
768	5,00	41,052	8,645	31,700	6,685	917	77,219%	2,518	3,032	6,429	6,268	52,005	53,503%
768	6,25	33,466	7,043	29,058	6,121	430	86,830%	2,388	2,919	6,153	6,899	47,354	48,718%
768	7,50	27,829	5,861	26,523	5,588	129	95,304%	1,379	2,146	4,147	7,411	42,833	44,067%
768	8,75	23,296	4,905	22,977	4,838	32	98,631%	0,691	1,518	2,650	7,383	36,158	37,200%
768	10,00	20,458	4,305	19,646	4,135	79	96,031%	0,556	1,452	2,428	7,088	30,223	31,094%
1024	1,25	222,117	46,785	35,850	7,603	18477	16,140%	4,652	3,323	8,939	2,606	80,995	83,328%
1024	2,50	111,334	23,456	39,115	8,285	7144	35,134%	3,521	3,296	7,773	3,840	69,828	71,839%
1024	3,75	74,340	15,656	35,683	7,546	3818	48,000%	3,233	3,312	7,505	4,594	62,230	64,023%
1024	5,00	55,363	11,661	34,777	7,347	2026	62,816%	2,855	3,091	6,843	5,440	57,927	59,596%
1024	6,25	44,759	9,425	32,221	6,798	1231	71,988%	2,520	3,014	6,408	5,999	53,264	54,799%
1024	7,50	37,021	7,799	30,023	6,332	684	81,099%	2,465	3,083	6,442	6,393	48,601	50,001%
1024	8,75	31,890	6,716	28,328	5,968	348	88,832%	2,356	2,831	6,008	7,128	47,346	48,710%
1024	10,00	27,403	5,769	25,849	5,444	152	94,327%	1,359	2,152	4,136	7,210	41,003	42,185%
2048	1,25	436,995	92,109	33,047	7,003	40160	7,562%	5,246	3,070	9,206	1,840	89,268	91,840%
2048	2,50	221,057	46,586	36,020	7,644	18338	16,294%	4,669	3,319	8,950	2,720	80,819	83,147%
2048	3,75	146,056	30,785	39,300	8,341	10575	26,907%	4,079	3,347	8,397	3,325	74,753	76,906%
2048	5,00	110,490	23,289	38,554	8,177	7118	34,893%	3,537	3,363	7,875	3,798	69,471	71,473%
2048	6,25	88,299	18,616	37,651	7,977	5007	42,640%	3,325	3,382	7,688	4,200	65,993	67,894%
2048	7,50	74,219	15,645	36,248	7,672	3750	48,840%	3,321	3,275	7,546	4,603	62,455	64,254%
2048	8,75	63,966	13,488	35,110	7,423	2845	54,889%	3,113	3,226	7,274	4,978	59,542	61,257%
2048	10,00	55,143	11,620	35,145	7,427	1969	63,735%	2,790	3,155	6,859	5,419	57,945	59,614%

Continuação Tabela A.5 – Resultados experimentais obtidos para o algoritmo TBs.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agntes (10 ³)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% Agentes	Overhead de Controle
768	1,25	3,350	32,984	3,147	0,203	15,293	0,069	0,126	3,393
768	2,50	1,663	17,633	1,562	0,101	15,914	0,070	0,129	1,814
768	3,75	1,099	11,878	1,033	0,066	16,094	0,070	0,128	1,222
768	5,00	0,819	8,947	0,773	0,046	16,125	0,069	0,129	0,921
768	6,25	0,667	7,435	0,629	0,038	16,273	0,069	0,129	0,765
768	7,50	0,554	6,170	0,523	0,031	16,325	0,070	0,130	0,635
768	8,75	0,463	0,000	0,437	0,026	15,994	0,067	0,128	0,000
768	10,00	0,406	4,178	0,385	0,020	15,597	0,065	0,123	0,430
1024	1,25	4,468	0,010	4,195	0,272	15,208	0,069	0,125	0,001
1024	2,50	2,235	23,552	2,100	0,135	15,825	0,070	0,128	2,423
1024	3,75	1,488	15,565	1,398	0,090	15,803	0,069	0,126	1,601
1024	5,00	1,105	11,840	1,039	0,066	15,986	0,069	0,128	1,218
1024	6,25	0,893	9,691	0,839	0,054	16,116	0,070	0,128	0,997
1024	7,50	0,738	7,904	0,695	0,043	15,879	0,068	0,127	0,813
1024	8,75	0,634	7,165	0,597	0,037	16,433	0,070	0,131	0,737
1024	10,00	0,544	0,000	0,514	0,030	15,988	0,068	0,126	0,000
2048	1,25	8,766	0,000	8,239	0,527	14,901	0,069	0,126	0,000
2048	2,50	4,429	0,035	4,160	0,268	15,235	0,069	0,126	0,004
2048	3,75	2,921	0,942	2,744	0,177	15,338	0,069	0,125	0,097
2048	5,00	2,209	22,705	2,075	0,133	15,608	0,069	0,126	2,336
2048	6,25	1,764	18,397	1,658	0,106	15,698	0,069	0,127	1,893
2048	7,50	1,481	15,583	1,391	0,090	15,805	0,069	0,127	1,603
2048	8,75	1,277	13,017	1,199	0,077	15,945	0,069	0,127	1,339
2048	10,00	1,099	11,352	1,032	0,067	15,850	0,069	0,127	1,168

Tabela A.6 – Resultados experimentais obtidos para o algoritmo TBu.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ¹⁰)	Largura de Banda Utilizada
128	1,25	27,644	5,817	27,555	5,798	0	99,677%	0,596	1,104	2,020	7,558	44,161	45,433%
128	2,50	14,039	2,928	14,038	2,928	0	99,990%	0,075	0,129	0,242	6,960	20,607	21,200%
128	3,75	9,369	1,954	9,368	1,954	0	99,996%	0,055	0,056	0,127	6,915	13,734	14,129%
128	5,00	6,953	1,448	6,953	1,448	0	99,996%	0,050	0,038	0,099	6,861	10,159	10,452%
128	6,25	5,597	1,167	5,597	1,167	0	99,998%	0,047	0,033	0,090	6,754	8,095	8,329%
128	7,50	4,705	0,982	4,705	0,982	0	99,997%	0,046	0,032	0,088	6,712	6,814	7,010%
128	8,75	3,972	0,827	3,972	0,827	0	99,992%	0,046	0,032	0,087	6,765	5,810	5,977%
128	10,00	3,423	0,713	3,423	0,713	0	99,995%	0,046	0,032	0,087	6,839	5,087	5,233%
256	1,25	55,485	11,641	35,468	7,463	198	63,924%	2,623	3,002	6,495	5,437	58,463	60,147%
256	2,50	27,811	5,834	27,794	5,831	0	99,940%	0,607	1,094	2,018	7,550	44,297	45,573%
256	3,75	18,723	3,930	18,722	3,930	0	99,995%	0,131	0,333	0,561	7,443	29,495	30,344%
256	5,00	13,941	0,029	13,940	2,924	0	99,995%	0,080	0,148	0,271	7,087	20,951	21,554%
256	6,25	11,206	2,350	11,205	2,350	0	99,992%	0,061	0,073	0,155	6,977	16,623	17,102%
256	7,50	9,461	1,984	9,460	1,983	0	99,995%	0,059	0,069	0,149	7,077	14,264	14,675%
256	8,75	7,969	1,672	7,969	1,671	0	99,995%	0,054	0,055	0,125	6,945	11,837	12,178%
256	10,00	6,997	1,468	6,997	1,468	0	99,996%	0,050	0,037	0,097	6,799	10,197	10,491%
512	1,25	111,035	23,350	38,242	8,075	7192	34,442%	4,030	3,213	8,175	3,828	80,224	82,535%
512	2,50	55,786	11,731	35,477	7,479	2006	63,595%	2,763	3,032	6,674	5,495	59,310	61,019%
512	3,75	37,058	7,794	30,353	6,389	0	81,907%	0,273	3,210	4,414	6,581	50,937	52,405%
512	5,00	27,911	5,873	27,789	5,848	0	99,562%	0,695	1,285	2,353	7,475	44,127	45,398%
512	6,25	22,617	4,755	22,570	4,745	0	99,792%	0,257	0,743	1,215	7,389	35,359	36,377%
512	7,50	18,438	3,877	18,434	3,876	0	99,980%	0,178	0,490	0,810	7,326	28,643	29,468%
512	8,75	16,102	3,387	16,100	3,387	0	99,984%	0,105	0,251	0,429	7,237	24,757	25,470%
512	10,00	13,636	2,868	13,636	2,868	0	99,993%	0,068	0,092	0,188	7,079	20,536	21,128%

Continuação Tabela A.6 – Resultados experimentais obtidos para o algoritmo TBu.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agtes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
128	1,25	2,583	28,397	2,442	0,141	16,131	0,067	0,127	2,921
128	2,50	2,583	24,933	2,462	0,122	15,138	0,056	0,109	2,565
128	3,75	2,584	24,069	2,459	0,124	14,902	0,052	0,103	2,476
128	5,00	2,584	24,029	2,461	0,123	14,910	0,050	0,099	2,472
128	6,25	2,584	23,482	2,459	0,125	14,736	0,048	0,095	2,416
128	7,50	2,583	23,121	2,458	0,125	14,634	0,047	0,093	2,379
128	8,75	2,584	23,463	2,458	0,126	14,747	0,045	0,089	2,414
128	10,00	2,584	23,286	2,458	0,125	14,669	0,045	0,088	2,396
256	1,25	2,584	27,768	2,431	0,152	16,012	0,068	0,126	2,857
256	2,50	2,584	28,514	2,446	0,137	16,176	0,066	0,125	2,934
256	3,75	2,584	27,391	2,456	0,127	15,861	0,061	0,118	2,818
256	5,00	2,584	25,382	2,463	0,120	15,272	0,056	0,110	2,611
256	6,25	2,583	24,955	2,459	0,125	15,165	0,054	0,106	2,567
256	7,50	2,584	24,899	2,461	0,122	15,156	0,053	0,104	2,562
256	8,75	2,584	24,605	2,460	0,124	15,058	0,051	0,100	2,531
256	10,00	2,584	23,767	2,459	0,125	14,836	0,050	0,100	2,445
512	1,25	2,584	51,754	1,560	1,023	24,173	0,095	0,190	5,324
512	2,50	2,583	27,856	2,433	0,151	16,043	0,068	0,127	2,866
512	3,75	2,584	28,491	2,438	0,146	16,197	0,067	0,126	2,931
512	5,00	2,584	28,342	2,443	0,141	16,109	0,066	0,125	2,916
512	6,25	2,584	27,133	2,451	0,132	15,757	0,063	0,120	2,792
512	7,50	2,584	26,435	2,455	0,129	15,564	0,060	0,115	2,720
512	8,75	2,584	25,900	2,459	0,124	15,424	0,059	0,113	2,665
512	10,00	2,584	25,297	2,460	0,124	15,241	0,056	0,110	2,603

Continuação Tabela A.6 – Resultados experimentais obtidos para o algoritmo TBu.

Tamanho da Sessão (10 ³)	MSIA	Mensagens Geradas (10 ⁵)	Bits Gerados (10 ⁶)	Mensagens Entruques (10 ⁵)	Bits Entruques (10 ⁶)	Mensagens Perdidas (10 ³)	Porcentagem em Entrega	Média do Atraso	Desvio Padrão do Atraso	Atraso Médio para 90%	Média de Saltos	Total de Bits Transmítidos (10 ⁶)	Largura de Banda Utilizada
768	1,25	166,934	35,140	35,396	7,494	13027	21,204%	4,807	3,237	8,982	2,897	84,268	86,696%
768	2,50	83,218	17,517	36,392	7,703	4635	43,730%	3,483	3,289	7,726	4,584	7,624	7,843%
768	3,75	55,505	11,689	35,696	7,539	1956	64,310%	2,720	3,079	6,691	5,541	58,665	60,355%
768	5,00	41,491	8,733	30,825	6,499	1050	74,293%	2,518	3,133	6,560	6,284	52,604	54,119%
768	6,25	33,688	7,092	31,508	6,636	0	93,527%	2,228	2,740	5,762	7,098	50,226	51,673%
768	7,50	27,634	5,817	27,432	5,775	0	99,268%	0,817	1,467	2,709	7,601	44,419	45,698%
768	8,75	23,706	4,992	23,645	4,979	0	99,745%	0,259	0,670	1,124	7,343	36,879	37,942%
768	10,00	21,252	4,475	21,200	4,465	0	99,756%	0,245	0,822	1,306	7,295	32,897	33,844%
1024	1,25	221,887	46,735	32,626	6,911	18761	14,704%	4,972	3,229	9,138	2,409	86,482	88,973%
1024	2,50	109,975	23,157	37,989	8,033	7114	34,544%	4,089	3,183	8,194	3,904	80,044	82,349%
1024	3,75	73,579	15,497	35,374	7,480	3774	48,076%	3,549	3,346	7,865	4,920	73,861	75,989%
1024	5,00	54,475	11,474	34,283	7,242	1992	62,932%	2,711	3,065	6,665	5,493	58,285	59,964%
1024	6,25	44,017	9,269	32,207	0,068	1158	73,168%	2,548	3,183	6,654	6,047	53,773	55,322%
1024	7,50	36,703	7,730	29,414	6,205	0	80,140%	2,697	3,296	6,948	6,572	50,147	51,592%
1024	8,75	32,441	6,834	30,668	6,461	0	94,536%	2,103	2,628	5,494	7,299	49,957	51,396%
1024	10,00	27,568	5,808	26,895	5,666	0	97,559%	0,843	1,649	2,970	7,502	43,510	44,763%
2048	1,25	437,487	92,216	26,855	5,688	40824	6,138%	5,632	2,839	9,294	1,665	92,139	94,793%
2048	2,50	219,450	46,259	32,413	6,866	18534	14,770%	4,957	3,268	9,173	2,460	86,307	88,794%
2048	3,75	147,046	30,993	35,930	7,620	10998	24,435%	4,571	3,261	8,778	3,183	82,569	84,948%
2048	5,00	110,457	23,283	35,671	7,559	7395	32,294%	4,132	3,290	8,376	3,893	79,542	81,833%
2048	6,25	88,582	18,672	36,476	7,728	516	41,177%	3,696	3,321	7,981	4,397	76,254	78,451%
2048	7,50	72,250	15,229	34,224	7,250	3758	47,369%	3,446	3,315	7,722	5,187	73,737	75,862%
2048	8,75	62,535	13,181	33,582	7,109	2861	53,701%	3,435	3,263	7,644	5,653	71,806	73,874%
2048	10,00	55,833	11,766	34,499	7,291	2104	61,789%	2,640	3,146	6,699	5,455	58,164	59,839%

Continuação Tabela A.6 – Resultados experimentais obtidos para o algoritmo TBu.

Tamanho da Sessão (10 ³)	MSIA	Agentes Gerados (10 ³)	Bits de Agtes (10 ⁶)	Agentes Recebidos (10 ³)	Agentes Perdidos (10 ³)	Número de Salto Médio dos Agentes	Vida Média dos Agentes	Vida Média dos 90% dos Agentes	Overhead de Controle
768	1,25	2,584	50,161	1,610	0,974	23,426	0,095	0,193	5,161
768	2,50	2,584	52,791	1,523	1,061	24,546	0,093	0,186	5,431
768	3,75	2,583	27,327	2,430	0,154	15,904	0,067	0,125	2,811
768	5,00	2,583	27,959	2,435	0,148	1,606	0,067	0,126	2,876
768	6,25	2,584	28,358	2,442	0,142	16,158	0,067	0,126	2,918
768	7,50	2,584	28,356	2,444	0,140	16,112	0,066	0,126	2,917
768	8,75	2,584	27,410	2,451	0,133	15,853	0,063	0,121	2,820
768	10,00	2,584	26,965	2,451	0,132	15,722	0,062	0,120	2,774
1024	1,25	2,583	50,058	1,617	0,966	23,464	0,096	0,195	5,150
1024	2,50	2,584	51,505	1,562	1,022	24,033	0,094	0,189	5,299
1024	3,75	2,584	53,091	1,504	1,080	24,716	0,092	0,184	5,462
1024	5,00	2,584	27,500	2,430	0,153	15,952	0,067	0,126	2,829
1024	6,25	2,584	28,147	2,433	0,150	16,106	0,067	0,127	2,896
1024	7,50	2,584	28,245	2,438	0,146	16,113	0,067	0,126	2,906
1024	8,75	2,584	28,609	2,440	0,144	16,216	0,067	0,127	2,943
1024	10,00	2,583	28,373	2,448	0,136	16,133	0,066	0,125	2,919
2048	1,25	2,584	48,174	1,681	0,903	22,958	0,096	0,193	4,956
2048	2,50	2,584	49,792	1,621	0,962	23,376	0,096	0,194	5,123
2048	3,75	2,584	51,021	1,584	0,999	23,775	0,095	0,192	5,249
2048	5,00	2,584	52,338	1,559	1,025	24,347	0,095	0,191	5,385
2048	6,25	2,584	52,987	1,531	1,053	24,696	0,093	0,185	5,451
2048	7,50	2,584	53,207	1,501	1,083	24,769	0,092	0,184	5,474
2048	8,75	2,584	52,535	1,501	1,082	24,350	0,090	0,180	5,405
2048	10,00	2,583	27,770	2,429	0,155	16,025	0,068	0,127	2,857

Referências

- [1] **COSTA, A.** *Analytic Modelling of Agent-Based Network Routing Algorithms*. 2002. 184p. Thesis (Doctor) - Faculty of Engineering, Computer and Mathematical Sciences, University of Adelaide. Australia. 2002.
- [2] **MOY, J. T.** *OSPF - Anatomy of an Internet Routing Protocol*. 8th printing. United States of America: Addison-Wesley, 2004. 345p.
- [3] **DI CARO, G.** *Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks*. Thesis (Doctor) - Faculté des Sciences Appliquées, Université Libre de Bruxelles. Bruxelles. 2004.
- [4] **SCHOONDERWOERD, R.; HOLLAND, O.; BRUTEN, J.; ROTHKRANTZ, L.** *Ant based load balancing in telecommunications networks*. *Adaptive Behavior*, 5(2): 169-207. 1996.
- [5] **DI CARO, G.; DORIGO, M.** *AntNet: Distributed Stimerbic Control for Communications Networks*. *Journal of Artificial Intelligence Research*, 9:317-365. 1998. URL: citeseer.ist.psu.edu/dicaro98antnet.html.
- [6] **DI CARO, G.** *A society of ant-like agents for adaptive routing in networks*. Thesis (Master) - Faculté des Sciences Appliquées, Université Libre de Bruxelles. Bruxelles. 2002.
- [7] **DI CARO, G.; DORIGO, M.** *AntNet: A mobile agents approuch to adaptive routing*. Technical Report IRIDIA 97-12 - IRIDIA, Université Libre de Bruxelles. Bruxelles.

1997.

- [8] **DI CARO, G.; DORIGO, M.** *Two ant colony algorithms for best-effort routing in datagram networks.* In Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98). 1998.
- [9] **DI CARO, G.; DORIGO, M.** *An Adaptative multi-agent routing algorithm inspired by ants behavior.* 1998.
- [10] **DI CARO, G.; DUCATELLE, F.; GAMBARDELLA, L. M.** *Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks.* Technical Report IDSIA-27-04-2004. Dalle Molle Institute for Artificial Intelligence. 2004.
- [11] **DI CARO, G.; DUCATELLE, F.; GAMBARDELLA, L. M.** *Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks.* In European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking. 2005.
- [12] **VARGA, A.** *Omnet++ version 3.01 User Manual.*
- [13] **GABBER, E.; SMITH, M.A.** *Trail Blazer: A routing algorithm inspired by Ants.* In Proceedings of the 12th IEEE (ICNP 04) International Conference on Network Protocols. 541-546p. 2004.
- [14] **FAROOQ, M.** *Implementação de um modelo de algoritmo baseado em formigas (AntNet-CL e AntNet-CO)*
- [15] **TULLOCH, M.** *Microsoft Encyclopedia of Networking.* Microsoft Press, Redmond, 2000.
- [16] **TANENBAUM, A.S.** *Computer Networks.* Prentice Hall, New Jersey, NY. quarta edição, 2003.
- [17] **LI, R.** *A Simulation System for Hierarchical Routing Using Ant Based Control.* Thesis (Master) - Faculty of Information Technology and Systems, Delft University of Technology. Netherlands, 2004.
- [18] **DIBOWSKI, H.** *Hierarchical Routing System using Ant Based Control.* Thesis (Master) - Faculty of Information Technology and Systems, Delft University of Technology. Netherlands, 2003.
- [19] **WHITE, T.** *Routing with Swarm Intelligence.* Technical Report SCE-97-15, sce, 1997.

- [20] **FLOYD, S.; PAXSON, V.** *Difficulties in simulating the internet.* IEEE/ACM Transaction on Networking. 2001.
- [21] **LATHI, P.B.** *Modern Digital and Analog Communication Systems.* Third Edition. Oxford University Press. 1998.
- [22] **FAROOQ, M. et al** *BEEhive an Energy-Aware Scheduling and Routing Framework.* Report. University of Dortmund..
- [23] **FAROOQ, M. et al** *BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior.* ANTS 2004, LNCS 3172.
- [24] **WOLF, T.** *Histórias da Floresta Encantada.* 1986.