Rodolfo da Silva Villaça

Hamming DHT e HCube: Arquiteturas distribuídas para busca por similaridade

Campinas
2013

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Rodolfo da Silva Villaça

Hamming DHT e HCube: Arquiteturas distribuídas para busca por similaridade

Tese de doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.
Área de concentração: Engenharia de Computação.

Orientador: Prof. Dr. Maurício Ferreira Magalhães

Este exemplar corresponde à versão final da tese defendida pelo aluno, e orientada pelo Prof. Dr. Maurício Ferreira Magalhães

Campinas
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

V711h

Villaça, Rodolfo da Silva, 1974-
Hamming *DHT* e *HCube*: Arquiteturas distribuídas para busca por similaridade / Rodolfo da Silva Villaça. – Campinas, SP : [s.n.], 2013.

Orientador: Maurício Ferreira Magalhães.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de computadores - Arquitetura. 2. Sistemas distribuídos. 3. Recuperação da informação. 4. Banco de dados - Busca. I. Magalhães, Maurício Ferreira,1951-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Hamming *DHT* and *HCub*e: Distributed architectures for similarity search
Palavras-chave em inglês:
Computer networks - Architectures
Distributed systems
Information retrieval
Database - Search
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora:
Maurício Ferreira Magalhães [Orientador]
Lasaro Jonas Camargos
Magnos Martinello
Edmundo Roberto Mauro Madeira
Eduardo Alves do Valle Junior
Data de defesa: 21-08-2013
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Rodolfo da Silva Villaça

Data da Defesa: 21 de agosto de 2013

Título da Tese: "Hamming DHT e Hcube: Arquiteturas Distribuídas para Busca por Similaridade"

Prof. Dr. Maurício Ferreira Magalhães (Presidente): _____

Prof. Dr. Lasaro Jonas Camargos: _____

Prof. Dr. Magnos Martinello: _____

Prof. Dr. Edmundo Roberto Mauro Madeira: _____

Prof. Dr. Eduardo Alves do Valle Junior: _____

Aos meus pais, Francis e Vera Villaça.

Agradecimentos

Agradeço,

Ao Prof. Maurício Magalhães, pela orientação, amizade e compreensão demonstrada durante todos esses anos de orientação.

A minha família, pela convivência diária, suporte emocional e todo o carinho e admiração que eu sei que eles têm por mim e eu tenho por eles.

À minha esposa, Leila, que durante alguns anos abdicou da minha presença e compreendeu a importância desta tese, dando-me força nos momentos mais difíceis que passei nessa jornada.

Aos colegas do LCA/FEEC/Unicamp: Rafael Pasquini, Luciano Ruivo, Christian, Walter, Daniel Botuca, Fabio Verdi, Jeremias, Carlos Macapuna, Jáder, dentre outros. A convivência com vocês correspondeu a um período muito gratificante de minha vida e certamente será lembrada para sempre.

Aos colegas que me ofereceram estadia gratuita em Campinas durante todo o tempo que fiquei ausente, em especial ao meu amigo José Eduardo

Aos colegas da Universidade Federal do Espírito Santo pela compreensão da importância dessa tese e por todo o apoio que me deram, cientes da dificuldade que é fazer todo esse trabalho estando mais de 1000km distante da Unicamp, sem afastamento e em exercício da docência.

À FEEC/UNICAMP pela ótima estrutura que oferece aos estudantes e pesquisadores. Certamente este será um modelo que eu perseguirei durante toda a minha vida acadêmica.

À CAPES e à Ericsson Research, pelo apoio financeiro no início dessa jornada e à FACOM/UFU, pela infra-estrutura disponibilizada para os testes com o HCube.

E nossa história não estará
pelo avesso assim,
sem final feliz.
Teremos coisas bonitas pra contar.

E até lá, vamos viver
Temos muito ainda por fazer.
Não olhe pra trás,
apenas começamos.
O mundo começa agora,
apenas começamos.

Metal Contra as Nuvens
Legião Urbana

Resumo

Atualmente, a quantidade de dados disponíveis na Internet supera a casa dos *Zettabytes* (ZB), definindo um cenário conhecido na literatura como *Big Data*. Embora as soluções de banco de dados tradicionais sejam eficientes na busca e recuperação de um conteúdo específico e exato, elas são ineficientes nesse cenário de *Big Data*, visto que não foram projetadas para isso. Outra dificuldade é que esses dados são essencialmente não-estruturados e encontram-se diluídos em toda a vastidão da Internet. Desta forma, novas soluções de infraestruturas de bancos de dados são necessárias de modo a suportar a busca e recuperação de dados similares de maneira não exata, configurando-se a busca por similaridade, isto é, busca por grupos de dados que compartilham entre si alguma semelhança. Nesse cenário, a proposta desta tese é explorar a similaridade de Hamming existente entre identificadores de objetos gerados através da função *Random Hyperplane Hashing*. Essa característica presente nesses identificadores servirá de base para propostas de infra-estruturas distribuídas de armazenamento de dados capazes de suportar eficientemente a busca por similaridade. Nesta tese serão apresentadas a Hamming DHT, uma solução P2P baseada em redes sobrepostas, e o HCube, uma solução baseada em servidores para *Data Center*. As avaliações de ambas as soluções são apresentadas e mostram que elas são capazes de reduzir as distâncias entre conteúdos similares em ambientes distribuídos, o que contribui para o aumento da cobertura em cenários de busca por similaridade.

Palavras-chave: Busca por similaridade, distância de Hamming, códigos de Gray, Tabelas Hash Distribuídas, Centros de Dados.

Abstract

Nowadays, the amount of data available on the Internet is over Zettabytes (ZB). Such condition defines a scenario known in the literature as Big Data. Although traditional database solutions are very efficient for finding and retrieving an specific content, they are inefficient on Big Data scenario, since the great majority of such data is unstructured and scattered across the Internet. In this way, new databases are required in order to support queries capable of finding and recovering similar datasets, i.e., retrieving groups of data that share a common meaning. In order to handle such challenging scenario, the proposal in this thesis is to explore the Hamming similarity existent between content identifiers that are generated using the Random Hyperplane Hashing function. Such identifiers provide the basis for building distributed infrastructures that facilitate the similarity search. In this thesis, we present two different approaches: a P2P solution named Hamming DHT, and a Data Center solution named HCube. Evaluations of both solutions are presented and indicate that such solutions are capable of reducing the distance between similar content, improving the recall in a similarity search.

Keywords: Similarity Search, Hamming distance, Gray codes, Random Hyperplane Hashing, Distributed Hash Tables, Datacenter.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ZB	ZettaBytes
P2P	Peer to Peer (Par a Par)
SFC	Space Filling Curve (Curva de Preenchimento de Espaço)
DHT	Distributed Hash Table (Tabela Hash Distribuída)
LSH	Locality Sensitive Hashing (Hashing Sensível à Localidade)
RHH	Random Hyperplane Hashing
API	Application Program Interface (Interface de Programação de Aplicação)
SRS	Social Recommender System (Sistema de Recomendação Social)
NN	Nearest Neighbors (Vizinhos Próximos)
k-NN	k-Nearest Neighbors (k-Vizinhos Próximos)
DNS	Domain Name System
SMTP	Simple Mail Transfer Protocol
MD5	Message-Digest Algorithm 5
SHA-1	Secure Hash Algorithm 1
TCP	Transmission Control Protocol
IP	Internet Protocol
DC	Data Center
COTS	Commodity Off-the-Shelf
lcp	longest common prefix
LSI	Latent Semantic Indexing
VSM	Vector Space Model
CBIR	Content-Based Image Retrieval
MIDI	Musical Instrument Digital Interface
IC	Intervalo de Confiança

Notações

n	Número de dimensões de um espaço vetorial
s	Ordem de uma SFC
sim	Função de Similaridade
sim_{cos}	Similaridade do Cosseno
d	Distância
d_{eud}	Distância Euclideana
sim_{eud}	Similaridade Euclideana
d_h	Distância de Hamming
sim_h	Similaridade de Hamming
d_{jac}	Distância de Jaccard
sim_{jac}	Similaridade de Jaccard
m	Comprimento, em bits, de uma cadeia binária
k	Índice, Chave ou Identificador
s	Ordem de uma curva de preenchimento de espaços
Pr	Probabilidade
$N(0,1)$	Distribuição Normal, com média 0 e variância 1
p	Identificador de um nó qualquer em uma DHT
N	Número de nós em uma DHT ou servidores em um Data Center
si	Identificador do servidor de admissão no HCube
di	Identificador dos dados de referência em uma consulta no HCube

Sumário

Introdução

Atualmente os usuários das redes de computadores tornaram-se ricas fontes de dados, interagindo entre si através de celulares, câmeras fotográficas ou *tablets*, publicando fotos, vídeos, músicas e documentos nas redes sociais, redes de compartilhamento de arquivo, *blogs* e comunidades virtuais. As empresas armazenam incontáveis informacões sobre seus clientes, provenientes de ferramentas de mineração de dados, sistemas de recomendação, serviços de localização e perfis de relacionamento em redes sociais. Milhões de sensores monitoram o mundo real, criam e trocam dados na "Internet das Coisas" (Berners-Lee, Hendler & Lassila 2001). De acordo com estudos da IDC (*International Data Corporation*) (Gantz & Reinsel 2010) e (Gens 2013), o universo digital irá expandir-se em mais de 50% em 2013, atingindo a marca de 4ZB (*ZettaBytes*). Os estudos também revelaram que cerca de 90% destes dados são não-estruturados, heterogêneos e variáveis em sua natureza, tais como textos, imagens e vídeos.

Essa quantidade de dados tão grande e tão diversificada é conhecida como *Big Data* e vem trazendo grandes desafios para os administradores de redes e para as empresas especializadas em armazenamento, distribuição e processamento de dados. Esses desafios estão relacionados à organização dos dados em estruturas de armazenamento, locais ou distribuídas, e à disponibilização de serviços de busca e recuperação de dados em larga escala. A análise e o processamento desses dados, com o objetivo de extrair informações úteis aos negócios dos clientes e à experiência dos usuários, também são desafios relevantes nesse cenário.

A busca por similaridade, também conhecida como busca pelos vizinhos mais próximos (*Nearest Neighbors Search*, NN) apresenta-se, nesse cenário, como uma alternativa capaz de viabilizar o armazenamento, distribuição, recuperação e processamento de grandes quantidades de dados. O termo "busca por similaridade" surgiu em 1973 com Donald Knuth (Knuth 1973) ao referir-se ao problema de posicionamento dos postos de correio mais próximos às residências dos seus clientes (*the post-office problem*). Ao longo dos anos, a busca por similaridade foi estendida a vários problemas nas diversas áreas da Ciência da Computação, chegando às redes Par a Par (*Peer to Peer*, P2P) de compartilhamento de arquivos (Bhattacharya, Kashyap & Parthasarathy 2005), (Villaça, de Paula, Pasquini & Magalhães 2013), (Haghani, Michel & Aberer 2009) e centros de dados (*Data Center*) (Park, Park & Lee 2011), (Wang, Chen, Huang, Xu, Yang & Xiao 2011a), (Villaça, Pasquini, de Paula & Magalhães 2013a).

Apesar das diferentes definições e aplicações conhecidas na literatura, este trabalho define a busca por similaridade como um tipo de busca na qual, a partir de uma consulta em uma base

de dados, o objetivo é recuperar todos os dados similares segundo um índice de similaridade requerido. Em outras palavras, trata-se de um tipo de busca que se propõe a responder à seguinte pergunta: "Quais são os objetos mais similares a um determinado objeto de referência?". Na literatura encontram-se muitos trabalhos relacionados a este tipo de busca que permitiram, ao longo dos anos, o desenvolvimento de diversas classes de aplicações. Dentre essas aplicações podemos citar: sistemas de recomendação (Lops, de Gemmis & Semeraro 2011), sistemas de auxílio ao diagnóstico por imagem (Petri 2011), máquinas de busca (Berry, Drmac, Elizabeth & Jessup 1999), sistemas reconhecimento de voz (Ryynanen & Klapuri 2008), etc.

Uma questão muito importante nesses trabalhos é a definição de similaridade. O que torna um objeto similar a outro em termos computacionais? Em geral, usa-se alguma forma de representação para os objetos (imagens, texto, áudio, vídeo, etc) de uma base de dados. Dentre diversas formas podemos destacar a representação vetorial. Nessa representação, várias características desses objetos compõem dimensões de um vetor n-dimensional e a similaridade entre eles é medida através de funções de distância vetorial, tais como a distância Euclideana, o cosseno do ângulo entre os vetores (Qian, Sural, Gu & Pramanik 2004) e a distância de Hamming (Liu, Shen & Torng 2011). Cada uma dessas medidas de distância possui características próprias que serão discutidas mais adiante no Capítulo 1, e cabe às aplicações de busca por similaridade optar por uma ou outra medida.

A partir da estimativa de similaridade entre os vetores que representam os conteúdos, pode-se gerar os índices que irão viabilizar a sua indexação em estruturas de armazenamento e, posteriormente, permitir a busca e recuperação por similaridade. Uma abordagem bastante utilizada é baseada na utilização de curvas de preenchimento de espaço (*Space Filling Curve*, SFC) para agregação de objetos similares e geração dos índices. Dentre esses trabalhos podemos citar (Liao, Lopez & Leutenegger 2001), (Lawder 1999), (Angiulli & Pizzuti 2002).

Lawder (Lawder 1999) fez um estudo de diferentes SFCs e concluiu que a curva de Hilbert apresenta a melhor agregação em estruturas de armazenamento baseadas na distância Euclideana segundo a sequência dos números naturais (conjunto $\mathbb{N}$). Entretanto, essa mesma curva apresenta o maior custo computacional para geração dos índices, uma vez que geralmente são empregados algoritmos recursivos cuja complexidade é proporcional ao número de dimensões do vetor. Em sua tese, Lawder apresenta a curva Z como aquela que possui a menor complexidade na geração dos índices, porém com a menor agregação no espaço Euclideano. A curva de Gray é também apresentada como aquela que possui uma agregação inferior à curva de Hilbert no espaço Euclideano, porém com menor custo computacional. Ainda referenciando-se à tese de Jonathan Lawder, a curva de Gray é adequada para a agregação na distância de Hamming, cujo significado corresponde ao número de posições nas quais duas strings de mesmo comprimento diferem entre si. Ainda em 1998 Lawder dizia *"um interesse considerável vem sendo apresentado na curva de Gray, mas não conhecemos nenhuma implementação prática que faça uso dessa curva"* [1]. Mesmo após 15 anos dessa afirmação, ainda há poucos trabalhos disponíveis na literatura que exploram a curva de Gray e suas propriedades. Dentre esses podemos citar (Bhattacharya et al. 2005) que apresenta uma extensão às DHTs (*Distributed Hash Table*) tradicionais, tais

[1] "considerable interest has been shown in the Gray-code curve but we are not aware of any practical implementations which utilizes this curve"

como Chord (Stoica, Morris, Liben-Nowell, Karger, Kaashoek, Dabek & Balakrishnan 2003) e Pastry (Rowstron & Druschel 2001), explorando as propriedades da distância de Hamming na busca por similaridade.

Em 1988, Faloutsos (Faloutsos 1988) demonstrou, analiticamente, que a utilização de códigos de Gray na ordenação numérica de blocos de discos rígidos reduzia o esforço físico na realização de buscas por intervalos (*range queries*) baseadas na distância de Hamming. O código de Gray organiza os identificadores em uma sequência tal que dois identificadores sucessivos diferem entre si em um único *bit* e a curva de Gray é, naturalmente, consequência espacial da ordenação gerada pelo código de Gray. O trabalho de Falousos é o primeiro do qual temos conhecimento que explora essa característica de agregação da curva de Gray em alguma estrutura de armazenamento de dados.

Em 1998 Indyk and Motwani (Gionis, Indyk & Motwani 1999) e (Indyk & Motwani 1998) apresentam o problema da maldição da dimensionalidade (*Curse of Dimensionality*), na qual a complexidade de geração dos índices usando uma SFC é proporcional ao número de dimensões do vetor. Em resumo, quanto mais características de um objeto forem consideradas, maiores serão as dificuldades para indexá-las visando a busca por similaridade. Em seus trabalhos eles propuseram o uso de funções de *hash* Sensíveis à Localidade (*Locallity Sensitive Hashing*, LSH) como uma alternativa para indexação de vetores multidimensionais. As funções LSH realizam a redução das dimensões de maneira probabilística na qual dados similares são mapeados nos mesmos índices, ou em índices próximos, com alta probabilidade. As funções LSH passaram a ser empregadas em trabalhos relacionados à busca por similaridade como uma alternativa às SFCs. Dentre esses trabalhos podemos citar (Haghani et al. 2009) e (Zhang, Xiao, Tang & Tang 2011).

Dentre as várias funções LSH existentes na literatura destaca-se a função RHH (*Random Hyperplane Hashing*) (Charikar 2002), uma família de funções LSH cuja similaridade corresponde ao cosseno do ângulo entre vetores em uma representação vetorial. Em (de Paula, Villaça & Magalhães 2011) mostrou-se experimentalmente que no uso de funções RHH para geração de índices, quanto maior for o grau de similaridade do cosseno entre os vetores, maior será a probabilidade de que os índices gerados possuam *bits* em comum, o que implica em identificadores com pequenas distâncias de Hamming.

Nesse contexto, diante do cenário exposto no início desta seção, detectou-se a oportunidade de aplicação dos conhecimentos já solidificados na literatura relacionados à busca por similaridade em ambientes distribuídos que permitam o armazenamento, busca, recuperação e processamento de dados em larga escala, tais como redes P2P e *Data Center*. Percebeu-se durante a revisão bibliográfica que a característica de preservação da similaridade do cosseno de vetores de objetos nos identificadores gerados por funções RHH era pouco explorada, tornando-se o principal motivador desta tese.

Hipóteses do Trabalho e Proposta de Tese

A hipótese principal assumida nesta tese é que a preservação da similaridade do cosseno entre vetores na distância de Hamming de identificadores gerados por funções LSH é uma característica

importante e, se existir uma infraestrutura especializada para busca e recuperação de dados similares com essa características, a busca por similaridade será facilitada.

Em função dessa hipótese, esta tese se propõe a explorar essa característica e investigar estruturas distribuídas de armazenamento de dados que privilegiem a distância de Hamming e a busca por similaridade. No decorrer do texto serão apresentadas duas diferentes abordagens: a primeira, a Hamming DHT, é uma rede P2P sobreposta, organizada sob a forma de DHT, que privilegia o estabelecimento de contatos (*fingers*) entre os pares de acordo com a distância de Hamming entre os seus identificadores; a segunda, o HCube, é uma proposta de organização de um *Data Center* centrado nos servidores cujo objetivo principal é alcançar bons resultados de cobertura (*recall*) na busca por similaridade e melhorar a eficiência da busca através da aproximação física dos servidores de conteúdo, o quê, nas redes P2P sobrepostas, é obtido pela proximidade lógica no espaço virtual de identificação dos pares.

O desenvolvimento deste trabalho baseia-se nas seguintes premissas:

- A busca por similaridade é uma maneira importante de se facilitar a organização, busca, recuperação e processamento de dados em ambientes de *Big Data* uma vez que a organização de dados por similaridade facilita a movimentação de grandes volumes de dados no *Data Center*;

- Redes P2P sobrepostas e *Data Center* provêem uma infraestrutura distribuída adequada para a organização, busca, recuperação e processamento de dados em ambientes de *Big Data*;

A seguir são enumerados e descritos os objetivos gerais e específicos desta tese.

Objetivos Gerais e Específicos

Esta tese tem como objetivo principal aplicar a característica de preservação da similaridade do cosseno entre vetores de objetos na distância de Hamming dos identificadores gerados por funções RHH, de tal forma a viabilizar a realização de buscas por similaridade nos objetos representados por esses vetores e mostrar que essa característica pode ser usada em aplicações centralizadas ou distribuídas de maneira eficiente, com resultados medidos pela cobertura (*recall*) das consultas realizadas. Neste trabalho, o termo "eficiência" refere-se ao número de saltos, que se traduz em latência na troca de informações em sistemas distribuídos. A cobertura, assim como a precisão, são as medidas fundamentais para avaliação da eficácia de sistemas de Recuperação de Informação (Berry et al. 1999).

São objetivos específicos desta tese:

1. Propor uma organização de rede P2P sobreposta (*Overlay*) que privilegie a realização de buscas por similaridade;

2. Propor uma organização de servidores em um *Data Center* visando privilegiar buscas por similaridade.

Como opção de desenvolvimento desse trabalho, fez-se o uso da função de *hash* RHH implementada em (de Paula, Villaça & Magalhães 2010b) e modificada de acordo com as necessidades desta tese. Além disso, utilizou-se a base de dados de adultos disponível em (Frank & Asuncion 2010) na realização dos testes e experimentações. Esta base de dados contém informações sobre perfis de adultos dos EUA provenientes do censo populacional do ano de 1994. O desenvolvimento de protótipos que permitissem a medição da cobertura foi a estratégia adotada nesta tese para testar e avaliar o cumprimento dos objetivos propostos.

Contribuições da Tese

A percepção da manutenção da similaridade do cosseno entre vetores na distância de Hamming dos identificadores gerados por funções RHH não é nova e foi apresentada ainda em 1998 por Indyk and Motwani (Gionis et al. 1999), (Indyk & Motwani 1998). Entretanto, durante toda a fase de revisão da literatura realizada ao longo do desenvolvimento desta tese, observou-se que essa característica é pouco explorada pelos muitos trabalhos relacionados à área.

Nesse contexto, as duas principais contribuições desta tese são: a Hamming DHT, uma rede P2P sobreposta, baseada em uma infraestrutura de DHT capaz de viabilizar a execução de buscas por similaridade de maneira eficiente no espaço virtual de endereçamento; e o HCube, que representa uma infraestrutura de *Data Center* centrada em servidores voltados para a organização e busca de dados por similaridade.

A abordagem usada na Hamming DHT é única na literatura, uma vez que ela é especializada na busca por similaridade e explora a distância de Hamming entre os identificadores. Na Hamming DHT, os identificadores dos nós e objetos (conteúdos) são ordenados segundo o código de Gray, de tal forma que dois identificadores sucessivos representam conteúdos muito similares, uma vez que no código de Gray dois identificadores sucessivos diferem em um único bit. Essa característica, aliada ao uso da função RHH para geração dos identificadores de conteúdo, provoca o agrupamento de conteúdos similares em um mesmo nó da DHT. Além disso, explora-se a característica de distribuição espacial dos identificadores vizinhos no código de Gray para estabelecimento de contatos de longa distância (*fingers*) entre os nós.

O HCube, por sua vez, é uma proposta única na literatura pois explora a similaridade na distância de Hamming dos identificadores gerados pela função RHH na organização de servidores em um *Data Center*. As arquiteturas tradicionais privilegiam o processamento de dados paralelos e, em geral, buscam o espalhamento dos dados visando balanceamento de carga, ao contrário da proposta de arquitetura do HCube, que prevê o agrupamento de dados similares em detrimento do balanceamento de carga. Soluções como MapReduce (Dean & Ghemawat 2008) e Apache Hadoop® (The Apache Software Foundation 2013) provêem soluções eficientes para o processamento paralelo e armazenamento e movimentação de grandes volumes de dados. Entretanto, ambos não são especializados na busca por similaridade, embora possam ser utilizados para este fim. Nesse caso, a vantagem de uso do HCube está em facilitar a movimentação de grandes volumes de dados, uma vez que eles estarão agrupados por similaridade em servidores próximos no *Data Center*.

Por fim, esta tese introduz um método para a criação de vetores de conteúdo a partir de

características de objetos em qualquer domínio. É sabido que, em se tratando de informações discretas (ou não-numéricas), a noção de similaridade ou distância não é tão simples e direta como acontece com os atributos numéricos (Desai, Singh & Pudi 2011). Esse foi um dos principais desafios enfrentados no início do desenvolvimento desta tese e é consequência do fato de que não existe um ordenamento implícito quando os atributos são discretos ou não-numéricos. Por exemplo, o que é maior: a nacionalidade "brasileira" ou "japonesa"? Qualquer ordenamento nesse sentido é relativo ao domínio de aplicação do objeto representado, podendo mudar e provocar interpretações distintas em outros domínios. O método usado será apresentada no Capítulo 2 e tem a característica de poder ser facilmente estendido a qualquer tipo de objeto e livre de interpretações distintas em domínios distintos.

Organização da tese

O Capítulo 1 apresenta uma revisão dos conceitos principais usados nesta tese e tem o objetivo de familiarizar o leitor com os termos mais comuns, tais como busca por similaridade, similaridade do cosseno, vetores, distâncias, distância de Hamming, similaridade de Hamming, funções de *hash*, curvas de preenchimento de espaço. Esse capítulo é necessário pois há uma grande quantidade de conceitos pertencentes a diferentes áreas do conhecimento.

O Capítulo 2 apresenta a base de dados de adultos (Frank & Asuncion 2010), usada na avaliação de todas as implementações realizadas nesta tese, e a metodologia de criação dos vetores a partir das características discretas de um objeto. Além disso, o capítulo traz uma aplicação desses conceitos no desenvolvimento de um sistema de busca por similaridade aplicado na base de dados de adultos.

O Capítulo 3 apresenta a Hamming DHT, uma rede P2P especializada na recuperação de conteúdos por similaridade. Trata-se de uma proposta distribuída, baseada em redes sobrepostas, que aproxima logicamente os conteúdos similares. O capítulo traz a motivação para a proposta da Hamming DHT, os algoritmos de indexação de nós e conteúdos, as primitivas de armazenamento e recuperação de conteúdos por similaridade e o procedimento de estabelecimento de contatos distantes entre os nós da DHT. Além disso, o capítulo apresenta os testes e as avaliações dos resultados obtidos que comprovam a eficiência da solução proposta.

O Capítulo 4 aplica os conceitos de similaridade na distância de Hamming em um ambiente de *Data Center* aproximando fisicamente os conteúdos similares armazenados nos servidores. O capítulo apresenta as motivações e inspirações para a proposta, o procedimento de geração dos identificadores de servidores e conteúdos, o mapeamento entre servidores e conteúdos, os testes e os resultados obtidos, de tal forma a comprovar que o HCube é uma solução eficiente e eficaz para implementação da busca por similaridade em ambientes de *Data Center*.

O Capítulo 5 apresenta os principais trabalhos relacionados procurando situar esta tese no estado da arte da busca por similaridade em ambientes distribuídos. Fez-se a escolha por deixar essa discussão para o fim da tese devido à grande quantidade de diferentes conceitos envolvidos e a multiplicidade de soluções propostas.

Finalmente, o Capítulo 6 apresenta as considerações finais e conclui a tese, apontando propostas de trabalhos futuros.

Capítulo 1

Revisão Bibliográfica

Este capítulo tem como objetivo familiarizar o leitor com os conceitos básicos discutidos ao longo do texto. Nas próximas seções serão apresentados os conceitos de Similaridade, Busca por Similaridade, Curvas de Preenchimento de Espaço, funções de *hash* Sensíveis à Localidade, função *Random Hyperplane Hashing* e Código de Gray. Todos esses conceitos formam a base teórica indispensável à compreensão do conteúdo das propostas aqui apresentadas: Hamming DHT e HCube.

1.1 Similaridade

Na literatura encontram-se diferentes definições para o termo "similaridade". Por exemplo, na Matemática, em especial na Geometria, diz-se que dois objetos geométricos são similares se eles possuem a mesma forma, admitindo-se rotações, translações e espelhamentos (Cederberg 2001). Em Linguística Computacional, similaridade é uma métrica na qual documentos ou termos são comparados em função do seu significado semântico (Lin 1998). Na Psicologia, similaridade é uma métrica de proximidade entre diferentes mapeamentos mentais (Tversky 1977).

Independente das diferentes definições, esta tese considera que uma função de similaridade sim é uma função que tem como objetivo medir o grau de semelhança entre dois objetos (a,b) em um domínio de conhecimento $\mathcal{D}$. O resultado da aplicação desta função está no intervalo real $\mathbb{R} = [0..1]$, sendo que 0 significa que os objetos não possuem nenhuma semelhança e 1 significa que ambos possuem total semelhança. Formalmente:

$$\forall a, b \in \mathcal{D},\ sim(a, b) : \mathcal{D}x\mathcal{D} \rightarrow [0..1]$$

Em termos computacionais, uma função de similaridade implementa um algoritmo para mapear os objetos em algum ponto do intervalo [0..1]. Na literatura define-se também o termo "dissimilaridade", que representa a distância entre dois objetos de um domínio. Dados dois objetos (a,b) em um domínio de conhecimento $\mathcal{D}$, define-se dissimilaridade δ como uma medida de distância entre esses objetos que satisfaz às seguintes condições:

$$\forall\ a \in \mathcal{D} : \delta(a,a) = 0$$

$\forall\ a,b \in \mathcal{D} : \delta(a,b) \geq 0$
$\forall\ a,b \in \mathcal{D} : \delta(a,b) = \delta(b,a)$
$\forall\ a,b,c \in \mathcal{D} : \delta(a,c) \leq \delta(a,b) + \delta(b,c)$

Diferentes representações computacionais dos objetos de um domínio podem privilegiar diferentes medidas de distância entre os objetos, o que implica em diferentes funções de similaridade. Dentre as formas de representação mais comuns encontradas na literatura destacam-se a representação vetorial e a representação por conjuntos, detalhadas nas Seções 1.1.1 e 1.1.2, respectivamente.

1.1.1 Representação Vetorial

Neste tipo de representação os objetos são caracterizados através de vetores onde cada dimensão é mapeada em alguma característica dos objetos. Por exemplo, se o objeto for um documento, o vetor pode ser formado pelas palavras-chave ou termos mais importantes do documento (Berry et al. 1999); se o objeto for uma imagem, o vetor pode ser formado pelo histograma de cores desta imagem (Kong 2009) ou, ainda, por *tags* que descrevem essa imagem (Rasiwasia, Vasconcelos & Moreno 2006). Abordagem semelhante pode ser usada em vídeos (Smeaton, Wilkins, Worring, de Rooij, Chua & Luan 2008). Dados disponibilizados pelos perfis dos usuários de redes sociais podem ser usados para representá-los sob a notação vetorial (Nayak 2011).

Considere esse último exemplo e suponha uma rede social na qual os usuários atribuem notas a tópicos de seu interesse, tais como `Engenharia` e `Medicina`. Essas informações podem ser usadas para formar um vetor que caracteriza os usuários dessa rede social. A Figura 1.1 ilustra um exemplo nesse contexto. Nela, temos 4 usuários dessa rede social representados pelos vetores $\overrightarrow{USUARIO1}$, $\overrightarrow{USUARIO2}$, $\overrightarrow{USUARIO3}$ e $\overrightarrow{NOVO\_USUARIO}$. Da figura podemos descrever o USUARIO1 pela tupla (3, 7), onde cada coordenada representa o interesse em cada assunto, nesse caso 3 em `Engenharia`, eixo x, e 7 em `Medicina`, eixo y. Da mesma forma, o USUARIO2 é descrito pela tupla (9, 1), o USUARIO3 pela tupla (5, 5) e o NOVO_USUARIO pela tupla (6, 2).

A representação usada na Figura 1.1 é bastante simplificada, uma vez que com duas características tem-se somente duas dimensões no espaço. Quanto mais detalhes houver, no vetor, a respeito do objeto sendo representado, e quanto mais elaborada for a escolha das dimensões, melhor caracterizado o objeto estará (Berry et al. 1999), considerando sempre o problema da "maldição da dimensionalidade". Entretanto, o exemplo na Figura 1.1 será usada apenas como uma referência visual para a melhor compreensão das medidas de distância e similaridade que serão apresentadas a seguir.

Similaridade do Cosseno

Uma medida de similaridade comum na literatura é o cosseno do ângulo entre vetores em uma representação vetorial. Nessa abordagem, quanto maior for a similaridade entre dois vetores, menor será o ângulo entre eles e, consequentemente, maior será o cosseno. O cosseno é uma

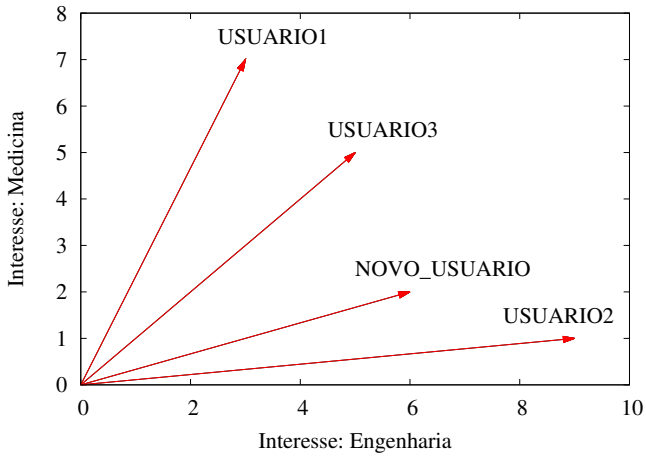


Figura 1.1: Espaço de interesses para USUARIO1, 2, 3 e NOVO_USUARIO.

medida de similaridade pois possui o seu resultado diretamente mapeado no intervalo [0..1]. Nos extremos, vetores coincidentes terão ângulo 0° e similaridade 1, e vetores ortogonais terão ângulo 90° e, consequentemente, similaridade 0.

Em geral, dado um par de vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ e um ângulo θ entre eles, pode-se calcular o cosseno através da seguinte fórmula:

$$\cos\theta(\overrightarrow{v_1},\overrightarrow{v_2}) = \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\|\|\overrightarrow{v_2}\|}$$

na qual $\overrightarrow{v_1} \cdot \overrightarrow{v_2}$ representa o produto escalar entre os vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ e $\|\overrightarrow{v_1}\|$ e $\|\overrightarrow{v_2}\|$ representam a norma desses vetores, respectivamente.

A partir dessa fórmula e considerando a Figura 1.1, o cosseno do ângulo entre o vetor $\overrightarrow{NOVO\_USUARIO}$(6, 2) e os vetores $\overrightarrow{USUARIO1}$(3, 7), $\overrightarrow{USUARIO2}$(9, 1), $\overrightarrow{USUARIO3}$(5, 5) pode ser calculado da seguinte forma:

$$\cos\theta(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO1}) = \frac{6*3+2*7}{\sqrt{6^2+2^2}*\sqrt{3^2+7^2}} = 0{,}664$$

$$\cos\theta(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO2}) = \frac{6*9+2*1}{\sqrt{6^2+2^2}*\sqrt{9^2+1^2}} = 0{,}977$$

$$\cos\theta(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO3}) = \frac{6*5+2*5}{\sqrt{6^2+2^2}*\sqrt{5^2+5^2}} = 0{,}894$$

Com estes resultados, usando o cosseno como medida de similaridade aplicada ao exemplo da Figura 1.1, pode-se afirmar que o USUARIO2 é o mais similar ao NOVO_USUARIO e, consequentemente, o USUARIO1 é o menos similar. Em (Lee, Park, Shim & Lee 2011) afirma-se que o cosseno é uma medida que produz bons resultados em diferentes tipos de dados e cenários de aplicação. Dessa forma, a similaridade do cosseno (sim_{cos}) entre dois vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ é a medida direta do cosseno entre eles:

$$sim_{cos}(\overrightarrow{v_1},\overrightarrow{v_2}) = \cos\theta(\overrightarrow{v_1},\overrightarrow{v_2})$$

Distância Euclideana

Ainda considerando-se a representação vetorial, uma outra forma medida de similaridade é a distância Euclideana entre os vetores. Nesse contexto, quanto menor for a distância entre eles, maior será a similaridade. Em um espaço vetorial n-dimensional, dados dois vetores $\overrightarrow{v_1}$ = $(v_{11}, v_{12}, v_{13}, ..., v_{1n})$ e $\overrightarrow{v_2} = (v_{21}, v_{22}, v_{23}, ..., v_{2n})$, a distância Euclideana d_{eud} é calculada da seguinte forma:

$$d_{eud}(\overrightarrow{v_1},\overrightarrow{v_2}) = \sqrt{(v_{11} - v_{21})^2 + (v_{12} - v_{22})^2 + (v_{13} - v_{23})^2 + ... + (v_{1n} - v_{2n})^2}$$

A partir desta fórmula, e usando-se os dados da Figura 1.1, as distâncias entre o vetor $\overrightarrow{NOVO\_USUARIO}$(6, 2) e os vetores $\overrightarrow{USUARIO1}$(3, 7), $\overrightarrow{USUARIO2}$(9, 1), $\overrightarrow{USUARIO3}$(5, 5) pode ser calculado da seguinte forma:

$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO1}) = \sqrt{(6-3)^2 + (2-7)^2} = 5{,}831$$
$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO2}) = \sqrt{(6-9)^2 + (2-1)^2} = 3{,}162$$
$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO3}) = \sqrt{(6-5)^2 + (2-5)^2} = 3{,}162$$

Com estes resultados, usando a distância Euclideana para medição da similaridade entre os vetores da Figura 1.1 pode-se afirmar que os usuários USUARIO2 e USUARIO3 são igualmente similares ao NOVO_USUARIO.

De acordo com (Qian et al. 2004), diferentes funções de similaridade produzem resultados distintos e a sua escolha depende fortemente do domínio de aplicação. Uma possível interpretação para esses resultados é tal que o USUARIO2 possui mais interesse em `Engenharia` do que em `Medicina`, entretanto em uma escala maior e com muito mais ênfase em `Engenharia`. O USUARIO3 possui um interesse equilibrado em ambas as dimensões, porém em uma escala mais próxima à escala de interesse do novo usuário. Essas diferentes interpretações podem fazer com que diferentes funções de similaridade sejam mais ou menos adequadas.

Uma característica importante da medição através da distância Euclideana é que ela não produz diretamente uma medida de similaridade no intervalo [0..1]. Para que essa medida de similaridade seja alcançada faz-se necessário uma normalização dos vetores nesse intervalo. Existem muitas técnicas de normalização para se atingir esse objetivo, dentre elas destaca-se uma técnica denominada Min-Max, na qual transforma-se um valor A em um valor B no intervalo $[C..D]$ através da seguinte fórmula:

$$B = \frac{A - min(A)}{max(A) - min(A)} * (D - C) + C$$

na qual $min(A)$ e $max(A)$ representam o menor e o maior valor possível para A, respectivamente.

No exemplo da Figura 1.1, considerando a normalização Min-Max no intervalo [0..1], 0 e 10 como o menor e o maior valor possível, respectivamente, a serem atribuídos aos tópicos `Enge-`

`nharia` e `Medicina`, os vetores normalizados seriam:

$$\overrightarrow{USUARIO1} = \left(\frac{3-0}{10-0}, \frac{7-0}{10-0}\right) = (0.3,\ 0.7)$$
$$\overrightarrow{USUARIO2} = \left(\frac{9-0}{10-0}, \frac{1-0}{10-0}\right) = (0.9,\ 0.1)$$
$$\overrightarrow{USUARIO3} = \left(\frac{5-0}{10-0}, \frac{5-0}{10-0}\right) = (0.5,\ 0.5)$$
$$\overrightarrow{NOVO\_USUARIO} = \left(\frac{6-0}{10-0}, \frac{2-0}{10-0}\right) = (0.6,\ 0.2)$$

As novas distâncias entre o vetor $\overrightarrow{NOVO\_USUARIO}$(0.6, 0.2) e os vetores $\overrightarrow{USUARIO1}$(0.3, 0.7), $\overrightarrow{USUARIO2}$(0.9, 0.1), $\overrightarrow{USUARIO3}$(0.5, 0.5), normalizadas no intervalo [0..1] são as seguintes:

$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO1}) = \sqrt{(0.6-0.3)^2 + (0.2-0.7)^2} = 0{,}583$$
$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO2}) = \sqrt{(0.6-0.9)^2 + (0.2-0.1)^2} = 0{,}316$$
$$d_{eud}(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO3}) = \sqrt{(0.6-0.5)^2 + (0.2-0.5)^2} = 0{,}316$$

Essas novas medidas no intervalo [0..1] representam a distância Euclideana entre os vetores e podem ser consideradas medidas de dissimilaridade. No exemplo, o USUARIO1 é o mais dissimilar ao NOVO_USUARIO por possuírem a maior distância entre si. A medida de similaridade Euclideana (sim_{eud}) entre dois vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ pode ser obtida da seguinte maneira:

$$sim_{eud}(\overrightarrow{v_1},\overrightarrow{v_2}) = 1 - d_{eud}(\overrightarrow{v_1},\overrightarrow{v_2})$$

Distância de Hamming

Uma terceira maneira de se medir a similaridade entre vetores n-dimensionais é através do cálculo da distância de Hamming entre eles (Kagie, Wezel & Groenen 2009). A distância de Hamming entre dois vetores é o número de dimensões nas quais eles diferem entre si. A distância normalizada no intervalo [0..1] pode ser obtida dividindo-se a distância de Hamming pelo número de dimensões n.

No exemplo da Figura 1.1 todos os pares possuem distância de Hamming igual a 2, pois não há coincidência em nenhuma dimensão em nenhum par. Entretanto, considere a existência de um vetor hipotético $\overrightarrow{USUARIO4}$(3, 2) no espaço da Figura 1.2. Esse vetor possui distância de Hamming 1 com o vetor $\overrightarrow{NOVO\_USUARIO}$(6, 2) pois coincidem os valores na dimensão `Medicina`.

As distâncias de Hamming (d_h) normalizadas, considerando n=2, são:

$$d_h(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO1}) = \frac{2}{2} = 1$$
$$d_h(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO2}) = \frac{2}{2} = 1$$

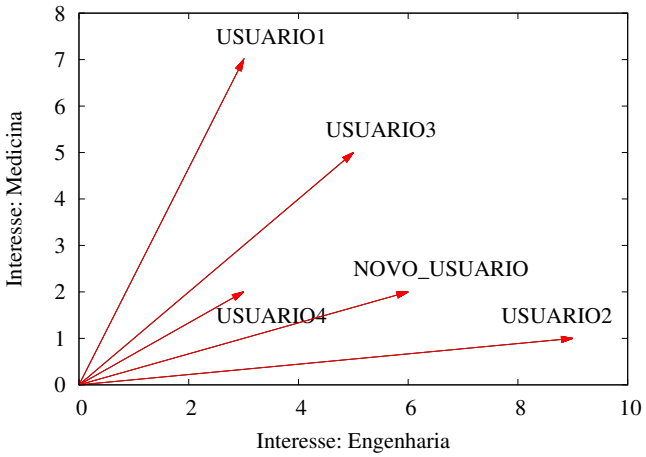


Figura 1.2: Inserção do USUARIO4 no espaço de vetores.

$$d_h(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO3}) = \frac{2}{2} = 1$$
$$d_h(\overrightarrow{NOVO\_USUARIO},\overrightarrow{USUARIO4}) = \frac{1}{2} = 0.5$$

Assim como a distância Euclideana, a distância de Hamming também é uma medida de dissimilaridade, portanto, quanto maior a distância, menor a similaridade. A similaridade de Hamming (sim_h) entre dois vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ pode ser calculada da seguinte maneira:

$$sim_h(\overrightarrow{v_1},\overrightarrow{v_2}) = 1 - d_h(\overrightarrow{v_1},\overrightarrow{v_2})$$

Para cadeias binárias define-se a distância de Hamming como sendo o número de posições de *bits* nas quais elas se diferenciam. A Tabela 1.1 mostra alguns exemplos de pares de cadeias binárias b_1 e b_2 de comprimento $m = 8$ bits. A mesma tabela também apresenta a similaridade de Hamming, que para cadeias binárias é calculada da seguinte forma:

$$sim_h(b_1,b_2) = 1 - d_h/m$$

onde d_h, nesse caso, representa a distância de Hamming normalizada.

Tabela 1.1: Distâncias e Similaridade de Hamming para pares de cadeias binárias b_1, b_2 e $m = 8$ *bits*.

b_1	b_2	d_h	sim_h
00100101	00100101	0	1
10100101	00110101	2	0.75
11100101	00111101	4	0.5
00100101	11011010	8	0

1.1.2 Representação por Conjuntos

Uma outra maneira de se representar objetos de uma maneira computacional é através do agrupamento das características e atributos que descrevem um objeto sob a forma de um conjunto. A principal diferença entre a descrição por conjunto e a descrição por vetor é que o primeiro é mais livre, não impondo restrições aos tipos e à ordem dos atributos que representam o objeto, enquanto o segundo possui restrições quanto aos valores em cada dimensão do vetor, que precisam ser numéricos. É possível transformar atributos não-numéricos em atributos numéricos e no Capítulo 2 será mostrada uma maneira de se realizar essa transformação.

Dois objetos muito similares possuirão uma grande intersecção entre os conjuntos que os representam. Dessa forma, quanto maior for a similaridade, maior será o conjunto interseção formado pelas características dos dois objetos. Nesse sentido, uma medida de similaridade muito comum entre dois conjuntos C_1 e C_2 é a Similaridade de Jaccard (Jaccard 1901) (sim_{jac}), definida como:

$$sim_{jac}(C_1,C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

onde $|C_1 \cap C_2|$ é o tamanho do conjunto interseção e $|C_1 \cup C_2|$ é o tamanho do conjunto união entre C_1 e C_2.

A distância de Jaccard (d_{jac}) mede a dissimilaridade entre dois conjuntos C_1 e C_2 através da seguinte fórmula:

$$d_{jac}(C_1,C_2) = \frac{|C_1 \cup C_2| - |C_1 \cap C_2|}{|C_1 \cup C_2|}$$

A Figura 1.3 mostra um exemplo com três conjuntos C_1, C_2 e C_3 que representam a relação dos elementos químicos que descrevem 3 objetos hipotéticos. $C_1 = \{$`Al`, `Hg`, `O`, `Au`, `Na`, `Cu`, `Cl`, `Ni`, `He`$\}$; $C_2 = \{$`Cu`, `Cl`, `Ni`, `He`, `H`, `C`, `F`, `U`, `Pb`$\}$ e $C_3 = \{$`Pb`, `Ar`, `Cs`, `Nd`, `Sg`, `Ga`, `P`$\}$.

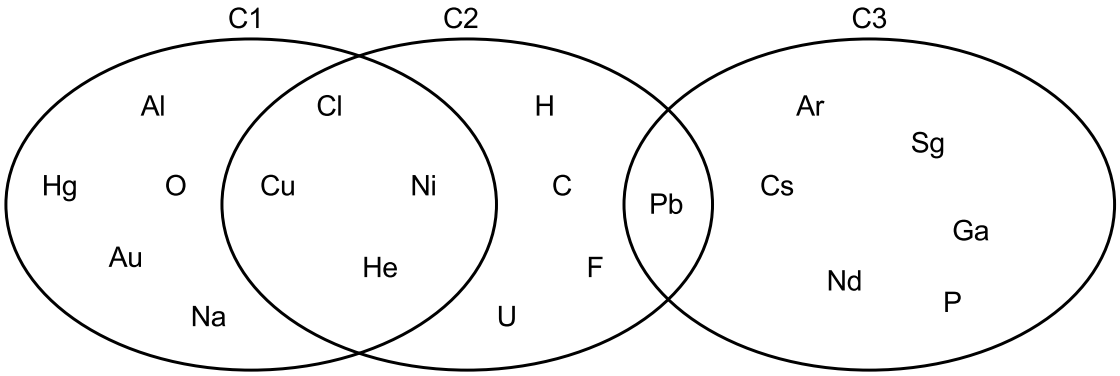


Figura 1.3: Conjuntos de elementos químicos C_1, C_2 e C_3.

A partir da Figura 1.3 é possível calcular:

$|C_1 \cap C_2| = |\{\texttt{Cu}, \texttt{Cl}, \texttt{Ni}, \texttt{He}\}| = 4$
$|C_1 \cap C_3| = |\{\}| = 0$
$|C_2 \cap C_3| = |\{\texttt{Pb}\}| = 1$

$|C_1 \cup C_2| = |\{\texttt{Al}, \texttt{Hg}, \texttt{O}, \texttt{Au}, \texttt{Na}, \texttt{Cu}, \texttt{Cl}, \texttt{Ni}, \texttt{He}, \texttt{H}, \texttt{C}, \texttt{F}, \texttt{U}, \texttt{Pb}\}| = 14$
$|C_1 \cup C_3| = |\{\texttt{Al}, \texttt{Hg}, \texttt{O}, \texttt{Au}, \texttt{Na}, \texttt{Cu}, \texttt{Cl}, \texttt{Ni}, \texttt{He}, \texttt{Pb}, \texttt{Ar}, \texttt{Cs}, \texttt{Nd}, \texttt{Sg}, \texttt{Ga}, \texttt{P}\}| = 16$
$|C_2 \cup C_3| = |\{\texttt{Cu}, \texttt{Cl}, \texttt{Ni}, \texttt{He}, \texttt{H}, \texttt{C}, \texttt{F}, \texttt{U}, \texttt{Pb}, \texttt{Ar}, \texttt{Cs}, \texttt{Nd}, \texttt{Sg}, \texttt{Ga}, \texttt{P}\}| = 15$

Logo, as similaridades de Jaccard (sim_{jac}) para os conjuntos C_1, C_2 e C_3 mostrados na Figura 1.3 são:

$$sim_{jac}(C_1,C_2) = \frac{4}{14} = 0.286$$
$$sim_{jac}(C_1,C_3) = \frac{0}{16} = 0$$
$$sim_{jac}(C_2,C_3) = \frac{1}{15} = 0.067$$

As respectivas dissimilaridades, medidas pela distância de Jaccard (d_{jac}) são:

$$d_{jac}(C_1,C_2) = \frac{14-4}{14} = 0.714$$
$$d_{jac}(C_1,C_3) = \frac{16-0}{16} = 1$$
$$d_{jac}(C_2,C_3) = \frac{15-1}{15} = 0.933$$

Em (de Paula et al. 2011) e (de Paula 2011) são apresentadas medidas e funções de similaridade aplicadas em modelagem conceitual através de ontologias. Entretanto, esses métodos não serão analisados por estarem fora do escopo desta tese por serem focadas na busca conceitual. Esta tese foca na representação vetorial voltada para a busca por similaridade pelo fato desta ser usada com bastante frequência na literatura relacionada e, também, por ser facilmente adaptada a diferentes domínios de aplicação devido a sua simplicidade de representação (Berry et al. 1999).

1.1.3 Busca por Similaridade

Nesta seção será apresentado o conceito de busca por similaridade, um dos principais elementos motivadores desta tese. Jagadish (Jagadish, Mendelzon & Milo 1995) define a busca por similaridade como sendo o problema de se encontrar, em uma coleção de objetos, aqueles que são mais similares a um objeto de referência usado na consulta (*query*).

As bases de dados tradicionais, sejam locais ou distribuídas, são eficientes na busca exata, na qual o objetivo é encontrar um objeto que satisfaz exatamente as características da consulta. Entretanto, a busca por similaridade em espaços multidimensionais é uma operação importante em diversas aplicações, tais como bases de dados multimídia (vídeo, áudio, imagens), sistemas de auxílio a diagnóstico médico por imagens, reconhecimento de objetos, bases de dados biológicos (sequências genéticas e de proteínas).

Os tipos de busca por similaridade mais comuns incluem a Busca por Intervalo (*Range Queries*) e a Busca pelos Vizinhos Próximos (*Nearest Neighbors*, NN). Na Busca por Intervalo são fornecidos intervalos de valores para algumas características dos objetos pesquisados e a consulta retornará aqueles cujas características estão contidas no intervalo fornecido. A Busca pelos Vizinhos Próximos caracteriza-se por receber um objeto de referência e algum valor de distância. A consulta deve retornar todos os objetos na coleção que estão próximos à referência de acordo com a distância requerida. A métrica de distância usada nesse tipo de busca depende fortemente da função de similaridade usada e da forma de representação dos objetos, conforme discutido na seção anterior. Há, ainda, uma variação da Busca pelos Vizinhos Próximos denominada "Busca pelos k-Vizinhos Próximos" (*k-Nearest Neighbors*, k-NN) (Cover & Hart 2006), (Haghani et al. 2009). Nesse tipo de busca, ao invés de se definir o valor da distância desejada, especifica-se um número k de objetos mais próximos que devem ser retornados após a consulta.

Para ser realizada eficientemente, a busca por similaridade requer o suporte de algum esquema de indexação. De acordo com (Valle & Cord 2009) as soluções para indexação de dados multidimensionais encontradas na literatura podem ser divididas em cinco classes:

- Soluções baseadas no particionamento de espaço, nas quais o hiperespaço é dividido em hiperretângulos mutuamente exclusivos;
- Soluções baseadas no particionamento e agrupamento dos dados, nas quais os dados são particionados em grupos mutuamente exclusivos, mas que podem se sobrepor no espaço;
- Soluções baseadas em quantização e aproximação, que reduzem os dados de tal forma a facilitar a busca sequencial pelos mesmos;
- Soluções baseadas em espaços métricos, que se utilizam de funções de distância para selecionar os dados a serem analisados;
- Soluções baseadas na projeção dos dados em subespaços, discretização dos mesmos e redução do espaço dimensional dos índices, agregando os resultados para facilitar a busca.

É consenso na literatura que a indexação de dados multidimensionais é uma tarefa árdua, que sofre da "maldição da dimensionalidade" ("*curse of dimensionality*") (Gionis et al. 1999), (Indyk & Motwani 1998). Em especial, as quatro primeiras soluções são afetadas por esse problema. A última classe de soluções é usada neste trabalho, em especial por evitar o problema da "maldição da dimensionalidade". A projeção em subespaços pode ser realizada de duas maneiras: através das curvas de preenchimento de espaço (*Space Filling Curves*, SFC) e através das Funções de *Hash* Sensíveis à Localidade (*Locality Sensitive Hashing*, LSH), sendo que essa última é menos afetada pela "maldição da dimensionalidade". Ambas serão detalhadas nas seções a seguir.

1.2 Curvas de Preenchimento de Espaço

As SFCs foram apresentadas pela primeira vez por Giuseppe Peano (Peano 1890) e podem ser definidas como curvas contínuas em um espaço n-dimensional que passam uma única vez

por cada um dos pontos desse espaço, ordenando-os. Relativamente ao exemplo da Figura 1.1, um espaço bi-dimensional, temos na Figura 1.4 uma SFC hipotética, traçada manualmente, passando por todos os pontos discretos do espaço de interesses (`Engenharia`, `Medicina`) que varia no intervalo de [1..12]. Nela, os pontos marcados em cinza correspondem à representação vetorial dos perfis USUARIO1(3, 7), USUARIO2(9, 1), USUARIO3(5, 5) e NOVO_USUARIO(6, 2).

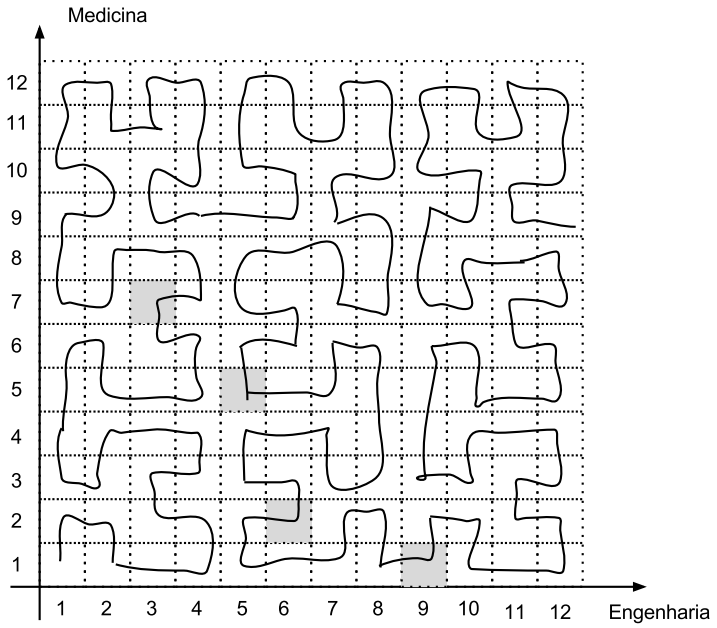


Figura 1.4: SFC hipotética no espaço de interesses bi-dimensional (`Engenharia`, `Medicina`).

A partir da SFC mostrada na Figura 1.4, ordenando-se os pontos do espaço (x,y) de acordo com a ordem com que são visitados, tem-se a geração de índices para os mesmos em um espaço unidimensional. Na mesma figura, com os índices iniciando-se em 0 (zero) e considerando um perfil hipotético USUARIO(1, 1), seu índice é $k = 0$. Seguindo a ordenação gerada pela curva, o índice do USUARIO1(3, 7) é $k_1 = 24$, do USUARIO3(5, 5) é $k_3 = 74$, do NOVO_USUARIO(6, 2) é $k_{new} = 88$ e do USUARIO2(9, 1) é $k_2 = 96$. Desta forma temos, portanto, uma discretização do espaço e uma redução do mesmo de 2 para 1 dimensão, conforme processo descrito na 5ª classe de soluções apresentadas na Seção 1.1.3.

Através dos índices gerados por essa SFC, pode-se dizer que o perfil mais similar ao NOVO_USUARIO é o USUARIO2, uma vez que ele possui a menor distância no espaço unidimensional gerado pela curva, no caso $d = |k_2 - k_{new}| = 8$. A Tabela 1.2 mostra as distâncias entre os índices dos perfis de usuários gerados pela curva da Figura 1.4 tomando-se o NOVO_USUARIO como referência. Na tabela, a distância é dada por $d = |k_{new} - k|$ e k corresponde aos índices dos usuários USUARIO1, USUARIO2 e USUARIO3, respectivamente.

Embora no exemplo tenha sido usado um espaço bi-dimensional para representar a curva, o mesmo conceito pode ser aplicado em espaços com um maior número de dimensões, sendo

Tabela 1.2: Distâncias entre os índices dos perfis dos usuários com relação ao índice do perfil NOVO_USUARIO.

Usuário	Índice	Usuário	Índice	Distância
NOVO_USUARIO	$k_{new} = 88$	USUARIO1	$k_1 = 24$	64
NOVO_USUARIO	$k_{new} = 88$	USUARIO2	$k_2 = 96$	8
NOVO_USUARIO	$k_{new} = 88$	USUARIO3	$k_3 = 74$	14

que a complexidade de geração da curva é diretamente proporcional ao número de dimensões do espaço, traduzindo-se na "maldição da dimensionalidade".

Nessa tese serão abordadas as SFCs de Hilbert, Z e Gray, relacionando-as à complexidade de geração das curvas e à agregação resultante das mesmas. As curvas apresentadas neste trabalho baseiam-se em espaços discretos, contendo um número finito de pontos e com a premissa básica de que pontos consecutivos nessas curvas também estão próximos no espaço n-dimensional. A primeira representação gráfica de uma SFC bidimensional é atribuída ao matemático David Hilbert (Hilbert 1891).

A Figura 1.5 mostra um processo recursivo de geração de uma curva de Hilbert de duas dimensões, onde as dimensões têm o mesmo número de pontos discretos no espaço. A primeira iteração produz uma curva com 2 pontos em cada dimensão; na segunda iteração dobra-se o número de pontos e, assim, sucessivamente. O número de iterações necessárias para geração da curva define a sua ordem.

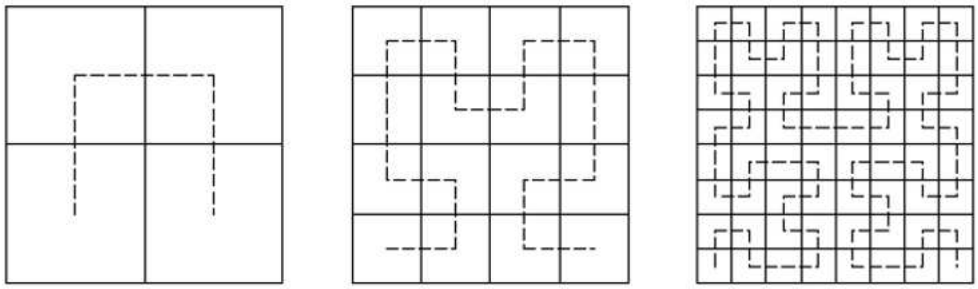


Figura 1.5: Representação gráfica da curva de Hilbert de 2 dimensões.

Algoritmos recursivos são frequentemente empregados na geração das curvas de Hilbert e no mapeamento dos pontos do espaço, mas abordagens interativas são mais eficientes. O custo computacional da geração da curva é proporcional ao número de dimensões n e à ordem da mesma. Lawder (Lawder 1999) apresenta um algoritmo para geração de curvas de Hilbert n-dimensionais de qualquer ordem. Segundo Lawder (Lawder 1999), algoritmos menos complexos podem ser usados se a ordem da curva gerada for uma potência de 2 (dois). Como exemplo podemos citar os algoritmos apresentados em (Butz 1971) e (Alber & Niedermeier 1998).

Entretanto, apesar da diversidade de abordagens existentes para geração das curvas, não existe um mapeamento direto, sem interações, entre um ponto do espaço e a sua posição relativa à curva, e vice-versa. Esse é o principal problema relativo ao custo computacional dos algoritmos

existentes.

As curvas de Hilbert são amplamente usadas em soluções de mapeamento de espaços multidimensionais e aplicadas na recuperação de conteúdos em diversas áreas porque apresentam a melhor agregação de índices no espaço unidimensional Euclideano, apesar do seu alto custo de geração.

Em 1966, Morton (Morton 1966) apresentou a curva Z através do mapeamento de pontos de um espaço bidimensional em uma curva unidimensional. O mapeamento das coordenadas do espaço n dimensional em sua posição na curva é feito de maneira direta e simples, intercalando-se ciclicamente os *bits* de cada coordenada para formar o índice da coordenada na curva. A Figura 1.6 mostra o processo de geração da curva Z em 3 iterações.

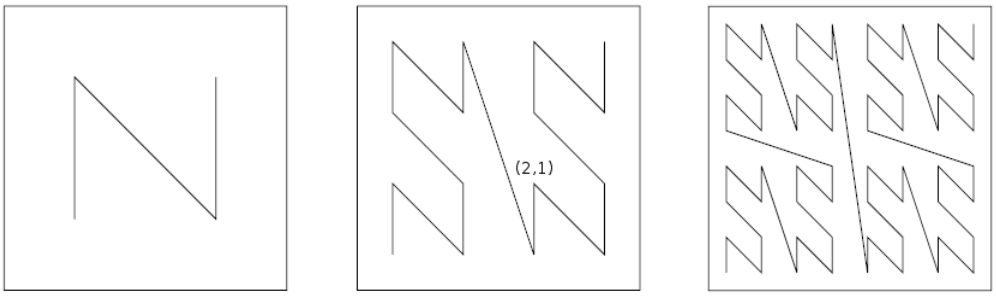


Figura 1.6: Representação gráfica da curva de Z de 2 dimensões.

Na Figura 1.6, tomando-se a primeira iteração como exemplo, tem-se 4 pontos, a saber: (0, 0), (0, 1), (1, 0), (1, 1). O mapeamento e ordenação destes 4 pontos é feito pela intercalação dos bits: 00 (0), 01 (1), 10 (2) e 11 (3). Na segunda iteração tem-se 16 pontos, a saber: (0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), ..., (3, 3). Como exemplo, o ponto (2,1), representado em binário (10_2, 01_2) é mapeado na posição 1001_2 (9) da curva.

A simplicidade do mapeamento nos pontos na curva Z torna-a uma curva com baixíssimo custo computacional. Considerando-se somente a definição estritamente matemática, a curva Z não é uma SFC, visto que não é contínua, i.e., pontos consecutivos na curva não são adjacentes no espaço. Como consequência ela agrega menos que a curva de Hilbert no espaço Euclideano. Porém, apesar da descontinuidade, vários trabalhos na literatura consideram a curva Z por sua simplicidade na geração dos índices. Lawder (Lawder 1999) avalia diversas curvas de preenchimento de espaço e também faz essa consideração.

Uma terceira SFC revisada nesta tese é a curva de Gray que, assim como a curva Z, também apresenta descontinuidades e, considerando a definição estritamente matemática, não é considerada uma SFC. Antes de apresentá-la, define-se um código de Gray como uma sequência de números inteiros binários na qual dois números consecutivos diferem entre si por somente um *bit*. O código de Gray foi apresentado por Gray (Gray 1953), aplicado na detecção de erros em comunicação de dados e será detalhado mais adiante nesta tese na Seção 1.4 por ser base do desenvolvimento das propostas aqui apresentadas.

A curva de Gray, apresentada em (Faloutsos 1988) e (Faloutsos 1986), oferece um compromisso entre a complexidade dos algoritmos de mapeamento da curva de Hilbert e a baixa agregação da curva Z. A Figura 1.7 mostra um exemplo de geração da curva de Gray em 4

(quatro) iterações, sendo que a 1ª iteração foi omitida por ser idêntica à 1ª iteração da curva de Hilbert. A segunda iteração está em destaque para ressaltar as coordenadas de acordo com a sequência do código de Gray, onde coordenadas consecutivas diferem em somente um *bit*.

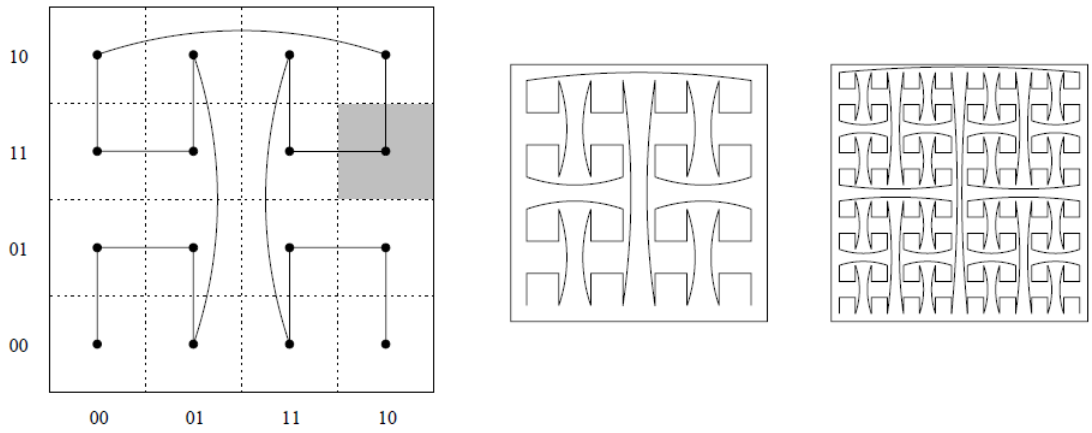


Figura 1.7: Representação gráfica da curva de Gray de 2 dimensões.

Na figura também está destacado com a cor cinza o ponto cujas coordenadas Gray são (10_2, 11_2) em binário, ou (2, 3) em decimal. Esse ponto está situado na posição 9 da curva de Gray, sendo vizinho das posições 8 e 10, onde se localizam os pontos cujas coordenadas são (10_2, 10_2) e (11_2, 10_2) em binário, ou (2, 2) e (3, 3) em decimal. Observe que os vizinhos diferem entre si em somente um *bit*. A característica desta curva, observada neste exemplo, é que a curva aproxima pontos na distância de Hamming e possui descontinuidades, o que, de acordo com (Valle, Cord & Philipp-Foliguet 2008), provoca uma perda na propriedade de agregação no espaço Euclideano quando comparada com a curva de Hilbert. Entretanto, conforme já definido, nesta tese estamos interessados na agregação na distância de Hamming.

Posteriormente, na Seção 1.4, é apresentado um algoritmo para geração de um código de Gray. Entretanto, já é possível adiantar que tais algoritmos são mais simples de serem implementados do que aqueles usados na geração da curva de Hilbert. Embora em todas as curvas apresentadas utilizamos somente cenários com 2 dimensões para facilitar a representação gráfica, todo o custo computacional de geração dessas curvas está relacionado à "maldição da dimensionalidade", na qual a complexidade dos algoritmos aumenta em função do número de dimensões do espaço.

1.3 Funções de *Hash* Sensíveis à Localidade

As funções de *Hash* Sensíveis à Localidade, ou LSH, são uma outra forma de redução de espaços n-dimensionais capaz de manter a proximidade dos pontos próximos neste espaço em um espaço de menor dimensão, geralmente unidimensional. Essas funções são usadas no contexto da indexação de dados e da busca por similaridade para criar identificadores de objetos em um sistema de busca de forma que a similaridade entre eles seja representada na distância entre os seus identificadores.

Em (Indyk & Motwani 1998) é definido que uma família de funções de *hash* $\mathcal{F}$ é classificada como sensível à localidade e correspondente a uma função de similaridade *sim*, se, para toda função $h \in \mathcal{F}$, temos:

$$Pr[h(a) = h(b)] = sim(a, b)$$

na qual a e b são objetos que possuem algum tipo de representação computacional e alguma função de similaridade associada a essa representação.

Existem vários tipos de funções LSH, sempre definidas de acordo com algum tipo de representação dos objetos e atreladas a alguma função de similaridade. A seguir são apresentados dois exemplos de função LSH: a primeira foi desenvolvida para objetos representados por conjuntos e usa a similaridade de Jaccard (função *Min-wise Hashing*); a segunda foi desenvolvida para objetos representados por vetores e usa a similaridade do cosseno (função *Random Hyperplane Hashing*).

1.3.1 *Min-wise*

As funções de *hash* baseadas em permutações independentes (*Min-wise Independent Permutations*) (Broder, Charikar, Frieze & Mitzenmacher 1998) são uma família de funções LSH voltadas para geração de índices de objetos representados por conjuntos de inteiros e comparados entre si através da similaridade de Jaccard (Seção 1.1.2).

Basicamente, a função de *hash Min-Wise* define uma permutação aleatória de *bits* a ser executada em todos os elementos do conjunto. O menor valor resultante dessas permutações será o resultado da função de *hash* e representará o identificador desse conjunto. Quanto maior a intersecção entre dois conjuntos, maior a chance de que os elementos da intersecção gerem o mesmo menor valor e, logo, possuam o mesmo identificador.

Formalmente, a função é definida da seguinte forma: seja π uma permutação binária aleatória no universo de inteiros $\mathbb{I}$; $C_1 = \{c_{11}, c_{12}, ..., c_{1i}\} \subseteq \mathbb{I}$, e $C_2 = \{c_{21}, c_{22}, ..., c_{2j}\} \subseteq \mathbb{I}$; a função de *hash* h_π é definida como:

$$h_\pi(C_1) = min\{\pi(c_{11}), \pi(c_{12}), ..., \pi(c_{1i})\}$$

onde $h_\pi(C_1)$ aplica a permutação π em cada elemento de C_1 e seleciona o menor dos resultados. Para dois conjuntos C_1 e C_2, temos que $x = h_\pi(C_1) = h_\pi(C_2)$ somente se $\pi^{-1}(x) \in (C_1 \cap C_2)$. Portanto, para dois conjuntos C_1 e C_2:

$$\mathbf{Pr}[h_\pi(C_1) = h_\pi(C_2)] = sim_{jac}(C_1,C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Em (Gupta, Gupta, Agrawal & Abbadi 2002) é mostrada uma implementação dessa função, exemplificada a seguir:

Dado um conjunto C de inteiros de m *bits*. associa-se um inteiro de referência que tenha exatos $\frac{m}{2}$ *bits* aleatórios iguais a 1. Para cada inteiro c pertencente ao conjunto C, move-se todo *bit* de c que corresponda às posições iguais a 1 do inteiro de referência para a metade superior de um novo inteiro c'. Os outros *bits* são movidos para a metade inferior. Esse

processo corresponde à 1ª iteração descrita na Figura 1.8. Em seguida, cria-se um novo inteiro de referência com $\frac{m}{2}$ *bits*, tendo exatos $\frac{m}{4}$ *bits* aleatórios iguais a 1. Novamente, move-se todo *bit* de c' que corresponda às posições iguais a 1 do inteiro de referência com $\frac{m}{2}$ *bits* para a metade superior de um novo inteiro c'' e os outros para a metade inferior. Esse processo corresponde à 2ª iteração descrita na Figura 1.8.

Esse procedimento é repetido até que o inteiro de referência gerado tenha tamanho igual a 2 *bits*, o que corresponde à 3^a iteração no exemplo da Figura 1.8. O resultado da permutação sobre o inteiro c é o inteiro c'''.

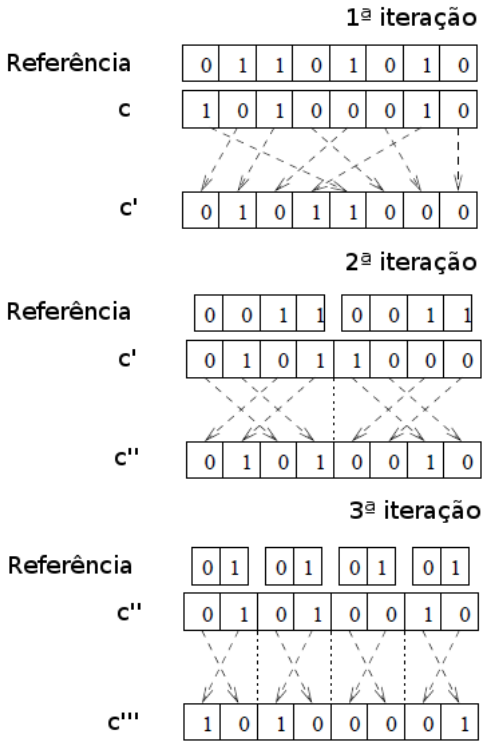


Figura 1.8: Exemplo de permutação *Min-Wise* para o inteiro c=151 de 8 *bits*. c''' é o resultado final da permutação.

Esse procedimento é refeito para todos os inteiros pertencentes ao conjunto C. Os mesmos inteiros de referência devem ser usados em todos os elementos do conjunto C. O menor valor gerado em todas essas permutações sobre todos elementos do conjunto será escolhido como o identificador desse conjunto. Deve-se usar os mesmos valores de referência para todos os conjuntos que serão indexados com uso de uma mesma função de *hash Min-Wise*.

1.3.2 *Random Hyperplane Hashing*

Dado um vetor de conteúdo $\overrightarrow{u} \in \Re^n$ que representa um objeto qualquer e um conjunto de m vetores $\overrightarrow{r}_i \in \Re^n$ $(1 \leq i \leq m)$ no qual cada coordenada desses vetores $\overrightarrow{r}_i$ é selecionada aleatoriamente a partir de uma distribuição normal padrão, a função *Random Hyperplane Hashing*,

ou RHH, $h_{\overrightarrow{r}_i}$ é definida como (Charikar 2002):

$$h_{\overrightarrow{r}_i}(\overrightarrow{u}) = \begin{cases} 1, \text{se } \overrightarrow{r}_i \cdot \overrightarrow{u} \geq 0 \\ 0, \text{se } \overrightarrow{r}_i \cdot \overrightarrow{u} < 0 \end{cases}$$

Para cada $h_{\overrightarrow{r}_i}(\overrightarrow{u})$ $(1 \leq i \leq m)$ um *bit* é gerado, e os m resultados da aplicação de $h_{\overrightarrow{r}_i}(\overrightarrow{u})$ são concatenados para compor um identificador de m *bits* para o vetor $\overrightarrow{u}$.

As funções RHH, formam uma família de funções LSH na qual os objetos são representados por vetores e, dados dois vetores de conteúdo $\overrightarrow{u}$ e $\overrightarrow{v} \in \Re^n$, que representam dois objetos quaisquer em um mesmo domínio, a função RHH $h_{\overrightarrow{r}_i}$ apresenta a seguinte característica (Charikar 2002):

$$\mathbf{Pr}[h_{\overrightarrow{r}_i}(\overrightarrow{u}) = h_{\overrightarrow{r}_i}(\overrightarrow{v})] = 1 - \frac{\theta(\overrightarrow{u},\overrightarrow{v})}{\pi} \approx \cos\theta(\overrightarrow{u},\overrightarrow{v})$$

(Charikar 2002) mostra que quanto menor for o ângulo $\theta(\overrightarrow{u},\overrightarrow{v})$ entre os vetores $\overrightarrow{u}$ e $\overrightarrow{v}$, mais essa probabilidade se aproximará do valor do cosseno entre eles $(\cos\theta(\overrightarrow{u},\overrightarrow{v}))$.

Dessa forma pode-se afirmar que quanto maior o cosseno entre os vetores, mais similares serão os objetos representados por eles. Essa relação entre a representação de similaridade da função RHH e o cosseno entre os vetores será referenciada como similaridade do cosseno, sim_{cos}, e quanto mais similares forem os objetos, melhor será a aproximação entre a similaridade e o cosseno entre os vetores.

Por consequência da concatenação da aplicação de m *bits* gerados pela aplicação de m funções $h_{\overrightarrow{r}_i}(\overrightarrow{u})$, para dois vetores de dados $\overrightarrow{u}, \overrightarrow{v} \in \Re^d$, quanto maior for a similaridade do cosseno entre esses vetores, maior será, com alta probabilidade, a similaridade de Hamming entre os identificadores gerados.

Para exemplificar este processo, considere que $\overrightarrow{u_1}$, $\overrightarrow{u_2}$ e $\overrightarrow{u_3}$ são vetores de inteiros de dimensão 6 (seis) que representam objetos em algum domínio de aplicação:

$\overrightarrow{u_1}$ = <10, 5, 6, 1, 0, 2>
$\overrightarrow{u_2}$ = <3, 4, 2, 10, 7, 6>
$\overrightarrow{u_3}$ = <14, 12, 0, 13, 1, 0>

Sejam $\overrightarrow{r_0}$, $\overrightarrow{r_1}$, $\overrightarrow{r_2}$, $\overrightarrow{r_3}$, $\overrightarrow{r_4}$, $\overrightarrow{r_5}$, $\overrightarrow{r_6}$ e $\overrightarrow{r_7}$ oito vetores de 6 dimensões selecionados aleatoriamente a partir de uma distribuição normal $N(0,1)$:

$\overrightarrow{r_0}$ = <-0.1, -0.9, -0.6, 0.5, 0.5, -0.8>
$\overrightarrow{r_1}$ = <0.8, 0.7, -0.6, -0.5, 0, -0.8>
$\overrightarrow{r_2}$ = <0, 0.1, 0.6, -0.9, -0.7, -0.3>
$\overrightarrow{r_3}$ = <0.4, -0.5, 0.1, -0.8, 0.2, 0>
$\overrightarrow{r_4}$ = <-0.7, -0.8, -0.7, 0, 0.2, -0.9>
$\overrightarrow{r_5}$ = <0.1, 0.5, 0.4, -0.1, -0.7, 0.6>
$\overrightarrow{r_6}$ = <-0.5, 0.5, 0.1, -0.7, 0.4, 0.3>
$\overrightarrow{r_7}$ = <0.4, 0.3, -0.2, 0, 0.1, -0.5>

São usados oito vetores $\overrightarrow{r}_i$ pois nesse exemplo serão gerados identificadores de $m = 8$ *bits* para os vetores $\overrightarrow{u_1}$, $\overrightarrow{u_2}$ e $\overrightarrow{u_3}$. É importante lembrar que cada vetor $\overrightarrow{r}_i$ provoca a geração de um único *bit*, e estes *bits* deverão ser concatenados para gerar o identificador do vetor. A Tabela 1.3 mostra o resultado do produto escalar entre o vetor $\overrightarrow{u_1}$ e cada um dos oito vetores $\overrightarrow{r}$. A tabela também mostra o resultado da aplicação da função $h_{\overrightarrow{r}}(\overrightarrow{u_1})$. Os resultados são concatenados e formam o identificador do vetor $\overrightarrow{u_1}$. $\overrightarrow{r_0}$ é o vetor que gera o *bit* de menor ordem. Desta forma $h_{\overrightarrow{r}}(\overrightarrow{u_1}) = 10101110_2$.

Tabela 1.3: Aplicação da função RHH no vetor $\overrightarrow{u_1}$.

Produto Escalar	$h_{\overrightarrow{r}}(\overrightarrow{u})$
$\overrightarrow{u_1} \cdot \overrightarrow{r_0}$ = -10.2	0
$\overrightarrow{u_1} \cdot \overrightarrow{r_1}$ = 5.8	1
$\overrightarrow{u_1} \cdot \overrightarrow{r_2}$ = 2.6	1
$\overrightarrow{u_1} \cdot \overrightarrow{r_3}$ = 1.3	1
$\overrightarrow{u_1} \cdot \overrightarrow{r_4}$ = -17	0
$\overrightarrow{u_1} \cdot \overrightarrow{r_5}$ = 7	1
$\overrightarrow{u_1} \cdot \overrightarrow{r_6}$ = -2	0
$\overrightarrow{u_1} \cdot \overrightarrow{r_7}$ = 3.3	1

As Tabelas 1.4 e 1.5 mostram os resultados da aplicação de $h_{\overrightarrow{r}_i}(\overrightarrow{u_2})$ e $h_{\overrightarrow{r}_i}(\overrightarrow{u_3})$, cujos resultados são 00100000_2 e 10100010_2, respectivamente.

Tabela 1.4: Aplicação da função RHH no vetor $\overrightarrow{u_2}$.

Produto Escalar	$h_{\overrightarrow{r}_i}(\overrightarrow{u})$
$\overrightarrow{u_2} \cdot \overrightarrow{r_0}$ = -1.4	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_1}$ = -5.8	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_2}$ = -14.1	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_3}$ = -7.2	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_4}$ = -10.7	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_5}$ = 0.8	1
$\overrightarrow{u_2} \cdot \overrightarrow{r_6}$ = -1.7	0
$\overrightarrow{u_2} \cdot \overrightarrow{r_7}$ = -0.3	0

Analisando os resultados da aplicação da função RHH no exemplo desta seção é possível comparar as similaridades do cosseno (sim_{cos}) entre os vetores $\overrightarrow{u}_i$ ($1 \leq i \leq 3$) e a similaridade de Hamming (sim_h) entre os identificadores gerados para cada vetor. A Tabela 1.6 sumariza esses resultados.

Tabela 1.5: Aplicação da função RHH no vetor $\overrightarrow{u_3}$.

Produto Escalar	$h_{\overrightarrow{r}_i}(\overrightarrow{u})$
$\overrightarrow{u_3} \cdot \overrightarrow{r_0}$ = -5.2	0
$\overrightarrow{u_3} \cdot \overrightarrow{r_1}$ = 13.1	1
$\overrightarrow{u_3} \cdot \overrightarrow{r_2}$ = -11.2	0
$\overrightarrow{u_3} \cdot \overrightarrow{r_3}$ = -10.6	0
$\overrightarrow{u_3} \cdot \overrightarrow{r_4}$ = -19.2	0
$\overrightarrow{u_3} \cdot \overrightarrow{r_5}$ = 5.4	1
$\overrightarrow{u_3} \cdot \overrightarrow{r_6}$ = -9.7	0
$\overrightarrow{u_3} \cdot \overrightarrow{r_7}$ = 9.3	1

Tabela 1.6: Comparação entre as similaridades do cosseno entre os vetores e a similaridade de Hamming entre os seus respectivos identificadores gerados pela função RHH.

Vetores	sim_{cos}	sim_h
$\overrightarrow{u_1}$, $\overrightarrow{u_2}$	0.445	$4/8 = 0.5$
$\overrightarrow{u_1}$, $\overrightarrow{u_3}$	0.732	$6/8 = 0.75$
$\overrightarrow{u_2}$, $\overrightarrow{u_3}$	0.687	$6/8 = 0.75$

De acordo com os resultados apresentados nessa tabela é possível notar que existe uma relação entre a similaridade do cosseno entre vetores e a similaridade de Hamming entre os identificadores destes vetores gerados por funções RHH. É importante destacar que a geração dos vetores $\overrightarrow{r}_i$ é aleatória e, portanto, os resultados podem variar de acordo com ela. Além disso, com identificadores de pequenos comprimentos (poucos *bits*) a similaridade de Hamming varia de maneira discreta com grandes intervalos de similaridade. Por exemplo, para $m = 8$ *bits*, os possíveis valores para a similaridade de Hamming são 0.0, 0.125, 0.25, 0.5, 0.625, 0.75, 0.875, 1. Isso ajuda a explicar por que os identificadores dos vetores $\overrightarrow{u_1}$ e $\overrightarrow{u_3}$ e $\overrightarrow{u_2}$ e $\overrightarrow{u_3}$ possuem a mesma similaridade de Hamming apesar de seus vetores não possuírem a mesma similaridade do cosseno.

Sem explicitar novamente todos os cálculos, apenas para efeito de ilustração e compreensão, a seguir, na Tabela 1.7, é apresentado o resultado de uma aplicação da função RHH nos vetores $\overrightarrow{u_1}$, $\overrightarrow{u_2}$ e $\overrightarrow{u_3}$ para geração de identificadores binários de 16 *bits*. Em seguida, a Tabela 1.8 traz a comparação entre a similaridade do cosseno e de Hamming entre os identificadores gerados. Como conclusão dos resultados apresentados nas Tabelas 1.7 e 1.8, é possível perceber que com o aumento do comprimento do identificador, a similaridade de Hamming entre $\overrightarrow{u_1}$ e $\overrightarrow{u_3}$, e entre $\overrightarrow{u_2}$ e $\overrightarrow{u_3}$ já consegue representar com maior precisão a similaridade do cosseno entre esses vetores.

Conforme já dito anteriormente, a função RHH é a escolha feita nesta tese, uma vez que foi feita a opção pela representação vetorial e similaridade do cosseno entre objetos. Além disso, essa função já fora usada em trabalhos anteriores deste grupo de pesquisa e estende os resultados e aplicações do trabalho desenvolvido em (de Paula 2011).

Tabela 1.7: Identificadores de 16 bits gerados por uma aplicação da função RHH.

Vetores	Identificador
$\overrightarrow{u_1}$	1011101011010101
$\overrightarrow{u_2}$	0111001110100111
$\overrightarrow{u_3}$	1011001111001101

Tabela 1.8: Comparação entre as similaridades do cosseno entre os vetores e a similaridade de Hamming entre os seus respectivos identificadores de 16 bits.

Vetores	sim_{cos}	sim_h
$\overrightarrow{u_1}$, $\overrightarrow{u_2}$	0.445	$8/16 = 0.5$
$\overrightarrow{u_1}$, $\overrightarrow{u_3}$	0.732	$12/16 = 0.75$
$\overrightarrow{u_2}$, $\overrightarrow{u_3}$	0.687	$10/16 = 0.625$

1.4 Códigos de Gray

Essa seção define e apresenta as propriedades dos códigos de Gray, no qual duas palavras binárias sucessivas diferem entre si em exatamente 1 *bit*, isto é, apresentam distância de Hamming igual a 1. Visto que a função RHH é a escolha desta tese, e que esta função gera identificadores de objetos cuja similaridade é preservada na distância de Hamming destes identificadores, é fácil notar que ordenando os identificadores segundo o código de Gray, identificadores vizinhos apresentarão, com alta probabilidade, uma grande similaridade entre si.

Um código de Gray binário de m *bits* (G_m) pode ser representado por uma matriz binária $2^m \times m$, onde cada linha é uma palavra deste código, que nesta tese representa um identificador (ou índice) de objeto. Os códigos de Gray são muitos (Suparta 2006) e, dentre eles, destaca-se o código de Gray binário refletido (*Binary Reflected Gray Code*). Esse código é o mais conhecido e utilizado na literatura por apresentar um algoritmo simples na sua construção. Esse algoritmo está explicado a seguir.

Dado que G_m é um código de Gray binário de m bits, então um código de Gray binário de $m+1$ bits (G_{m+1}) é produzido da seguinte forma: 1) concatena-se todos os identificadores do código G_m com os mesmos identificadores desse código na ordem reversa, construindo uma nova matriz G_{m+1}, com $2^{m+1} \times m$ identificadores; 2) adiciona-se o prefixo 0 aos identificadores da metade superior da matriz G_{m+1} e; 3) adiciona-se o prefixo 1 aos identificadores da metade inferior da matriz G_{m+1}. Ao fim desse procedimento está gerado o código G_{m+1}, com $2^{m+1} \times (m+1)$ identificadores.

Por indução, a aplicação recursiva desse método a partir do código de Gray trivial (G_1) (vide Figura 1.9) leva à construção de códigos de Gray binários de qualquer comprimento (G_m). A mesma figura ilustra a geração de G_2, G_3, G_n e G_{n+1} a partir de G_1.

A principal vantagem deste código de Gray com relação aos demais é que é possível converter diretamente uma sequência binária natural em um código de Gray binário refletido, e vice-versa. Essa conversão pode ser realizada usando operações simples e de baixo custo computacional, tais como OU-exclusivo, deslocamentos de *bits* e comparações com 0. Os algoritmos 1 e 2 formalizam

$$G_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad G_2 = \begin{bmatrix} 00 \\ 01 \\ 11 \\ 10 \end{bmatrix} \quad G_3 = \begin{bmatrix} 000 \\ 001 \\ 011 \\ 010 \\ 110 \\ 111 \\ 101 \\ 100 \end{bmatrix}$$

$$G_n = \begin{bmatrix} G_n[0] \\ G_n[1] \\ \dots \\ G_n[2^n - 1] \end{bmatrix} \quad G_{n+1} = \begin{bmatrix} 0G_n[0] \\ 0G_n[1] \\ \dots \\ 0G_n[2^n - 1] \\ 1G_n[2^n - 1] \\ \dots \\ 1G_n[1] \\ 1G_n[0] \end{bmatrix}$$

Figura 1.9: Geração de códigos de Gray binários refletidos G_2, G_3, G_n e G_{n+1} a partir de G_1.

essa conversão.

Algorithm 1: Conversão de Binário para Gray.

Input : Binary codeword, (*binary*).
Output: Gray codeword, (*gray*).
$gray \leftarrow binary \oplus (binary \gg 1)$;

No próximo capítulo, a base de dados de adultos (Frank & Asuncion 2010) usada na avaliação das soluções propostas nesta tese, será apresentada. Depois, nos Capítulos 3 e 4, são propostas duas soluções onde esses conceitos podem ser integrados para viabilizar a construção de estruturas distribuídas de armazenamento e recuperação de dados especializadas na busca por similaridade.

Algorithm 2: Conversão de Gray para Binário.

Input : Gray codeword, ($gray$).
Output: Binary codeword, ($binary$).
$binary \leftarrow gray$
$i \leftarrow (gray \gg 1)$
while $i \neq 0$ **do**
  $binary \leftarrow (binary \oplus i)$
  $i \leftarrow (i \gg 1)$
end

Capítulo 2

Busca por similaridade baseada na distância de Hamming

Com o objetivo de avaliar as propostas que serão apresentadas nesta tese foi necessário adotar uma base de dados de vetores que se adequasse às características desejadas. Nesse contexto foi utilizada a base de dados *Adult* (Frank & Asuncion 2010) da Universidade da Califórnia em Irvine (UCI), que apresenta dados de adultos extraídos do censo norte-americano de 1994. Esta base de dados será apresentada nas seções deste capítulo, junto com uma metodologia de criação de vetores numéricos a partir de dados com atributos textuais. Esse procedimento é necessário pois, conforme visto no Capítulo 1, as medidas de similaridade para vetores são extraídas a partir de funções que exigem atributos com valores numéricos em cada uma de suas dimensões.

Ainda neste capítulo, a função de *hash* RHH é aplicada aos vetores da base de dados e é feita uma análise dos identificadores gerados. São avaliados os resultados obtidos a partir da medição da correlação existente entre a similaridade do cosseno entre os vetores de adultos e a similaridade de Hamming dos identificadores gerados. A distribuição de frequência das distâncias de Hamming entre pares de identificadores em função da similaridade do cosseno entre os vetores também é avaliada. Finalizando, este capítulo apresenta uma proposta de um serviço genérico de busca por similaridade baseado na distância de Hamming dos identificadores dos vetores. O capítulo termina com uma breve análise da metodologia adotada e direcionamentos futuros voltados para uma implementação distribuída desse serviço.

2.1 A base de dados *Adult*

Conforme já dito, a base de dados *Adult* contém informações sobre cidadãos norte-americanos extraídas do censo populacional americano do ano de 1994. A base de dados é composta por 48842 vetores que descrevem os adultos em função de valores para os seus atributos. Traduzindo-os, os atributos disponíveis são: `idade`, `classe de trabalho`, `nível de escolaridade`, `tempo de escolaridade`, `estado civil`, `ocupação profissional`, `relacionamento`, `raça`, `sexo`, `ganho de capital`, `perda de capital`, `jornada de trabalho`, `nacionalidade`, `renda anual`. Algumas informações podem estar indisponíveis para alguns atributos em alguns vetores. Nesses casos, o atributo em questão está marcado com o símbolo de pontuação de

interrogação (?).

A seguir são apresentados alguns exemplos de vetores selecionados aleatoriamente na base de dados:

- Adulto 1 - 43; Self-emp-not-inc; 5th-6th; 3; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 4700; 20; United-States; <50K
- Adulto 2 - 56; Private; 10th; 6; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 4500; 30; Philippines; <50K
- Adulto 3 - 50; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; $\geq$50K
- Adulto 4 - 45; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
- Adulto 5 - 30; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 30; United-States; $\geq$50K

Nos arquivos de descrição da base de dados há a informação de todos os valores possíveis para cada um dos atributos do vetor. A seguir estão listados todos os atributos e seus respectivos valores:

- age: valor numérico.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-adm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education_num: valor numérico.
- marital_status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

- `sex`: `Female, Male`.
- `capital_gain`: valor numérico.
- `capital_loss`: valor numérico.
- `work_hours`: valor numérico.
- `native_country`: `United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands`.
- `annual_salary`: $\geq$`50K,<50K`.

2.2 Criação dos Vetores

Já fora comentado anteriormente que este trabalho foca na representação vetorial de objetos e geração dos índices dos vetores através da função RHH. É sabido que a função RHH usa o cosseno como métrica de similaridade entre vetores e, por consequência, é necessário que o vetor seja composto por atributos numéricos. Conforme apresentado em (Desai et al. 2011), esse procedimento é necessário pois a noção de distância ou similaridade para atributos textuais, ou categóricos segundo o autor, não é tão direta e usual quanto em dados numéricos. Esse foi um dos primeiros desafios encontrados no desenvolvimento desta tese.

O método desenvolvido não pode ser considerado uma contribuição desta tese. Entretanto, uma de suas principais características é que ele pode ser usado em uma enorme variedade de tipos de objetos descritos por vetores. Em (Bamba, Subercaze, Gravier, Benmira & Fontaine 2012) é apresentado um método mais elaborado para geração de vetores a partir do perfil de usuários na rede social Twitter®. Outras referências a respeito de técnicas mais apuradas para a criação de vetores podem ser encontradas em (Choi, Cha & Tappert 2010) e (Cha, Tappert & Yoon 2006). Como o objetivo não é explorar as diferentes formas de modelagem de dados, mas sim propor soluções de armazenamento distribuído voltadas para a busca por similaridade, optou-se por um método mais simples para geração dos vetores. Apesar da simplicidade ele produz bons resultados, os quais serão apresentados no decorrer deste capítulo.

Sabe-se que atributos textuais não se traduzem facilmente em alguma ordenação numérica. Isso acontece porque a noção de distância nessa ordenação é completamente dependente da interpretação no domínio da aplicação que faz uso destes dados. Por exemplo, suponha que seja atribuído o valor 1 para `female` e 2 para o `male` no atributo `sex` do vetor de adultos. Uma primeira aplicação pode interpretar estes valores somente como uma diferenciação entre os valores, uma espécie de índice para os mesmos. Entretanto, uma segunda aplicação pode entender que o valor `male` é o dobro do valor `female`, o que é uma verdade numérica mas

pode ser interpretada, dependendo do domínio, como uma afirmação "machista". Há, ainda, uma terceira categoria de dados não explorada nesta tese: a ordinal. Na categoria ordinal os atributos também são textuais, entretanto é possível atribuir uma ordem aos valores de tal forma a numerá-los. Um exemplo seria o grau de instrução de um adulto, onde quanto mais instruído, maior seria o valor numérico atribuído ao valor textual.

Entretanto, optou-se por não enveredar, nesta tese, pelo campo da modelagem de dados. Como solução simplificada, cada valor presente nos atributos do vetor de adultos gerou novos atributos em um novo vetor, expandindo-se assim o número de dimensões do vetor original. O objetivo principal desta transformação foi evitar que os valores de um mesmo atributo fossem relacionados entre si, por exemplo através de suas correlações, método proposto em (Le & Ho 2005). Com o procedimento proposto, o atributo `sex`, por exemplo, foi expandido gerando dois outros atributos no vetor expandido em sua substituição: `male` e `female`. Assim, seguiu-se o procedimento para os demais atributos do vetor original, exceto para aqueles que originalmente possuem valores numéricos e, portanto, permaneceram inalterados.

Após a realização dessa transformação para todos os atributos do vetor de adultos original, o vetor resultante expandido possui 106 dimensões, resultantes da expansão de valores dos 14 atributos originais. O vetor expandido possui os seguintes atributos, nessa ordem: `age, Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked, Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool, education_num, Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse, Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces, Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried, White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black, Female, Male, capital_gain, capital_loss, work_hours, United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands, ≥50K, <50K`.

O próximo passo na metodologia consistiu na atribuição de valores numéricos para cada atributo. O procedimento adotado nesta tese foi o seguinte: cada um dos atributos expandidos do vetor original possui valores discretos iguais a 0 ou 1, indicando a presença ou ausência daquele atributo no vetor resultante; para os atributos numéricos do vetor original, realizou-se a normalização Min-Max (apresentada no Capítulo 1 para o intervalo contínuo [0..1]). O objetivo do procedimento adotado neste segundo passo da transformação do vetor é não privilegiar nenhum atributo do vetor resultante, de tal forma que todos tenham, no máximo, o mesmo valor numérico.

Como exemplo, após a transformação, os 5 vetores de adultos apresentados nesta seção serão representados pelos vetores a seguir:

- Adulto 1 - `0.43, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0.166, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0.000, 0.944, 0.444, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1`

- Adulto 2 - `0.56, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0.333, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0.000, 0.900, 0.666, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1`

- Adulto 3 - `0.50, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.833, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0.000, 0.000, 0.800, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0`

- Adulto 4 - `0.45, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1.000, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0.000, 0.000, 0.889, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0`

- Adulto 5 - `0.30, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.833, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0.000, 0.000, 0.666, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0`

Quanto à normalização Min-Max dos atributos numéricos dos vetores, os limites utilizados foram [0..100] para o atributo `age`, [0..18] para o atributo `education_num`, [0..5000] para os atributos `capital_gain` e `capital_loss` e [0..45] para o atributo `work_hours`. Esses valores foram obtidos a partir da análise dos valores contidos nos vetores da base de dados utilizada.

A partir destes vetores modificados foi medida a similaridade do cosseno entre esses 5 adultos. Os resultados estão apresentados na Tabela 2.1:

A seguir são apresentadas algumas análises dos identificadores gerados pela aplicação da função RHH na geração de identificadores de 128 *bits* a partir dos vetores modificados.

Tabela 2.1: Cosseno entre os vetores modificados dos adultos 1, 2, 3, 4 e 5.

	Adulto 1	Adulto 2	Adulto 3	Adulto 4	Adulto 5
Adulto 1	1.0000	0.7095	0.5467	0.5442	0.5420
Adulto 2	0.7095	1.0000	0.4788	0.5722	0.5635
Adulto 3	0.5467	0.4788	1.0000	0.8122	0.9012
Adulto 4	0.5442	0.5722	0.8122	1.0000	0.9015
Adulto 5	0.5420	0.5635	0.9012	0.9015	1.0000

2.3 Avaliação da aplicação da função RHH nos vetores de adultos

A função de *hash* implementada pelo grupo da Unicamp e apresentada em (de Paula et al. 2010b) foi modificada para gerar identificadores a partir dos vetores de adultos. Em uma primeira avaliação, na Tabela 2.2 estão apresentadas as similaridades de Hamming para identificadores de 128 *bits* dos cinco vetores de adultos usados como exemplo no início deste capítulo e gerados usando a função RHH.

Tabela 2.2: Similaridade de Hamming entre os vetores modificados dos Adultos 1, 2, 3, 4 e 5.

	Adulto 1	Adulto 2	Adulto 3	Adulto 4	Adulto 5
Adulto 1	1.0000	0.7422	0.6172	0.6016	0.5859
Adulto 2	0.7422	1.0000	0.5078	0.6016	0.6094
Adulto 3	0.6172	0.5078	1.0000	0.8594	0.9218
Adulto 4	0.6016	0.6016	0.8594	1.0000	0.9375
Adulto 5	0.5859	0.6094	0.9218	0.9375	1.0000

A análise visual dos resultados apresentados na Tabela 2.2 mostra que a relação entre as similaridades se mantêm, embora os valores absolutos não sejam os mesmos, o que já era esperado, por dois motivos principais: 1) a similaridade de Hamming é uma grandeza discreta, cujos saltos entre os valores dependem do tamanho dos identificadores gerados - para $m = 128$ *bits*, o salto é $1/128 \approx 0.0078$; e 2) a função RHH é probabilística e os resultados dependem dos vetores $\overrightarrow{r}_i$, gerados aleatoriamente. É importante destacar neste ponto que esse conjunto de vetores $\overrightarrow{r}_i$, explicados na Seção 1.3.2 do Capítulo 1, deve ser mantido constante ao longo da geração de todos os identificadores de vetores em uma base de dados.

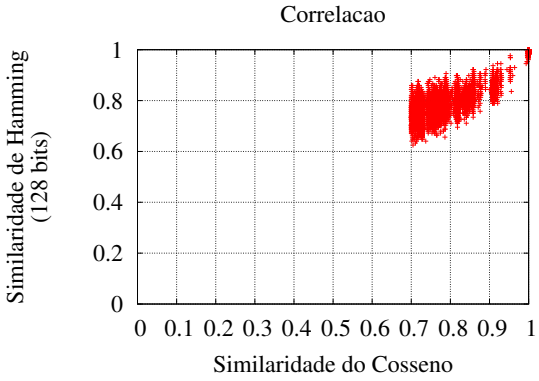
A qualidade dos resultados pode ser medida pela correlação entre o cosseno de pares de vetores de adultos e a similaridade de Hamming dos seus identificadores. O objetivo é avaliar se, realmente, a similaridade do cosseno entre os vetores é representada pela similaridade de Hamming dentre os seus respectivos identificadores. Sabe-se que correlação não implica causalidade, e portanto, mais adiante, na Seção 2.4, será feita uma análise qualitativa destes

resultados através de exemplos de buscas por similaridade baseada na distância de Hamming entre identificadores de vetores.

Com o objetivo de avaliar melhor a correlação, em função de diferentes graus de similaridade, os pares de vetores foram divididos em grupos, de acordo com a similaridade do cosseno entre eles. Os grupos selecionados foram:

- Grupo 1 - Pares de vetores de adultos cuja similaridade do cosseno é maior ou igual a 0.7 e menor que 0.8 ($0.7 \leq sim_{cos} < 0.8$);
- Grupo 2 - Pares de vetores de adultos cuja similaridade do cosseno é maior ou igual a 0.8 e menor que 0.9 ($0.8 \leq sim_{cos} < 0.9$);
- Grupo 3 - Pares de vetores de adultos cuja similaridade do cosseno é maior ou igual a 0.9 e menor que 0.95 ($0.9 \leq sim_{cos} < 0.95$);
- Grupo 4 - Pares de vetores de adultos cuja similaridade do cosseno é maior ou igual a 0.95 ($sim_{cos} \geq 0.95$);

A Figura 2.1 correlaciona a similaridade do cosseno entre pares de vetores de adultos dos grupos 1, 2, 3 e 4 com a similaridade de Hamming entre os seus respectivos identificadores. Por exemplo, a partir da figura é possível notar que, para similaridades do Grupo 1, a similaridade de Hamming correspondente varia no intervalo aproximado [0.62..0.85] e esse intervalo diminui para pares de vetores mais similares, o que mostra que a correlação entre as similaridades do cosseno e de Hamming aumenta quanto mais similares forem os vetores.



(a)

Figura 2.1: Relacionamento entre similaridade do cosseno e similaridade de Hamming entre pares de vetores dos grupos 1, 2, 3 e 4.

A Tabela 2.3 mostra o resultado numérico obtido para o coeficiente de correlação de Pearson entre as similaridades de pares de vetores dos grupos 1, 2, 3 e 4.

Tabela 2.3: Coeficiente de Correlação de Pearson entre as similaridades do cosseno e de Hamming para os grupos 1, 2, 3, 4.

Grupo	Coeficiente de Correlação de Pearson
1	0.84
2	0.90
3	0.95
4	0.98

Diferentes autores fazem diferentes interpretações a respeito dos valores do coeficiente de correlação de Pearson. Por exemplo, pode-se citar (Buda 2011) e (Cohen 1988). Entretanto, qualquer interpretação é influenciada pelas características do domínio da aplicação dos dados sob análise, mas praticamente todas elas concordam que correlações acima de 0.4 são consideradas moderadas e correlações fortes são aquelas maiores que 0.8 ou 0.9. Dessa forma, pelos resultados obtidos na Tabela 2.3, pode-se afirmar que existe uma correlação forte entre as similaridades, e esta correlação torna-se mais forte quanto mais similares forem os vetores.

A próxima avaliação é a distribuição de frequência das distâncias de Hamming entre os identificadores de vetores adultos gerados pela função RHH em função da similaridade do cosseno entre eles. Esse resultado está intimamente ligado ao anterior, uma vez que a similaridade de Hamming é obtida através da distância de Hamming entre identificadores, conforme visto no Capítulo 1. Para todos os pares de identificadores de vetores nos grupos 1, 2, 3 e 4 foi medida a distância de Hamming entre eles, contando-se o número de ocorrências em relação ao tamanho total dos grupos. Aqui cabe esclarecer que o tamanho total da base de dados de adultos é 48842 adultos, o que dá um total de $48842^2 = 2385540964$ pares. O Grupo 1 possui cerca de 10% destes pares, o Grupo 2 possui cerca de 4%, o Grupo 3 possui cerca de 2.5% e o Grupo 4 possui cerca de 1% deste total.

A Figura 2.2 apresenta esses resultados. Na análise foram usados os mesmos identificadores gerados para a análise dos resultados da correlação. Os identificadores têm comprimento m igual a 128 bits. Os pares de vetores de adultos foram novamente divididos nos grupos 1, 2, 3 e 4. Referindo-se ao Grupo 1 pode-se observar que as distâncias de Hamming variam na faixa de [18..47], o que corresponde a similaridades de Hamming iguais a [0.63..0.85], mostrando que os resultados são coerentes com os resultados de correlação obtidos na Figura 2.1. Análise semelhante pode ser feita para os outros 3 (três) grupos, representados nas Figuras 2.2(b)), 2.2(c) e 2.2(d).

Na próxima seção, é feita a proposta de um serviço de busca por similaridade baseado na distância de Hamming entre os identificadores dos vetores. Nela serão apresentados alguns resultados de consultas na base de dados de adultos, correspondendo a uma análise qualitativa dos mesmos.

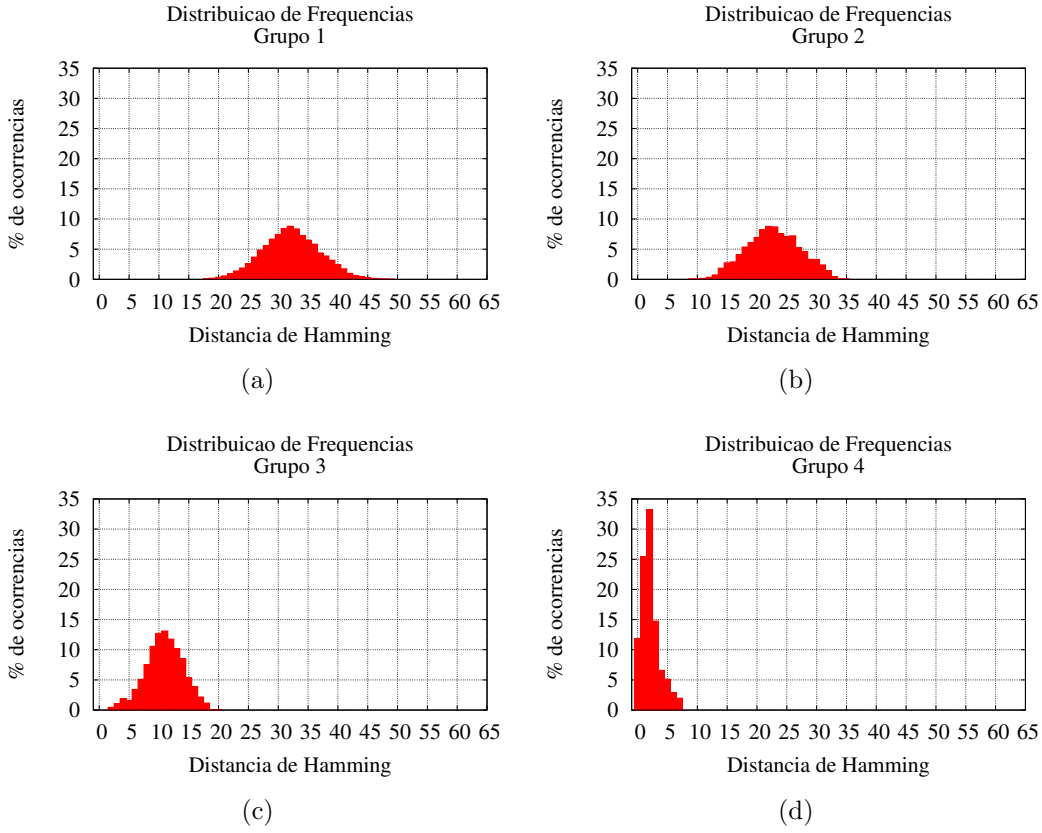


Figura 2.2: Distribuição de Frequência das Distâncias de Hamming para os grupos 1, 2, 3 e 4.

2.4 Um serviço de busca por similaridade baseado na similaridade de Hamming

Baseando-se nos resultados prévios da análise da correlação e distribuição de frequência, esta seção apresenta a proposta de um serviço de busca por similaridade baseado na similaridade de Hamming entre os identificadores dos vetores que representam objetos armazenados em uma base de dados local. O objetivo é avaliar a qualidade dos resultados das consultas em função das informações dos adultos recuperados pelo sistema.

Uma visão geral do funcionamento do sistema durante uma busca por similaridade é apresentada na Figura 2.3. Nela, o usuário monta um vetor original, descrevendo as características desejadas para o objeto procurado e um grau de similaridade de Hamming desejado. O sistema processa esse vetor, transformando-o em um vetor numérico, conforme procedimento descrito na Seção 2.2 e, a partir desse vetor, é gerado um valor de *hash* através da função RHH. Esse valor de *hash* corresponde ao identificador de um objeto "virtual", que não necessariamente está na base de dados, mas possui as características desejadas pelo usuário. Dessa forma esse mesmo valor de *hash* é usado como referência para buscar identificadores de objetos similares na base de dados. Nesse caso, todos os vetores já indexados na base de dados são comparados a esse valor com base em seus identificadores. Todos os objetos pertencentes à base de dados e que possuem similaridade de Hamming maior ou igual ao valor de referência são retornados como

resposta à consulta.

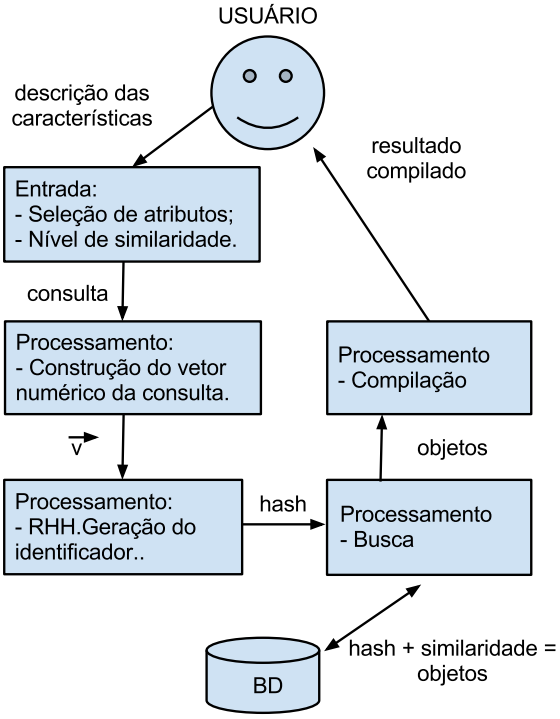


Figura 2.3: Funcionamento de um serviço de busca por similaridade baseado na distância de Hamming entre os identificadores.

Formalmente, para todo identificador i pertencente ao conjunto $\mathcal{S}$ de objetos similares ao identificador k, usado como referência na consulta, a similaridade de Hamming entre i e k deve ser maior ou igual ao nível de similaridade sim_h especificado:

$$\forall i \in \mathcal{S} : \frac{m - d_h(i,k)}{m} \geq sim_h$$

onde m é o comprimento (em *bits*) do identificador e $d_h(i,k)$ é a distância de Hamming entre duas cadeias binárias i e k.

Esse serviço foi implementado com o objetivo de possibilitar a avaliação qualitativa dos resultados das consultas. A seguir serão apresentados alguns resultados aleatórios extraídos de consultas baseadas nos Adultos 1, 2, 3, 4 e 5 usados como exemplo neste capítulo. Foram usados níveis de similaridade de Hamming maiores ou iguais a 0.7, 0.8, 0.9 e 0.95.

A Tabela 2.4 traz alguns resultados para consultas baseadas no Adulto 1 - `43`; `Self-emp-not-inc`; `5th-6th`; `3`; `Married-civ-spouse`; `Craft-repair`; `Husband`; `White`; `Male`; `0`; `4700`; `20`; `United-States`; `<50K` - e níveis de similaridade 0.7, 0.8 e 0.9. Não houve resultado para similaridade 0.95. Para similaridade 0.9 houve um único resultado.

A Tabela 2.5 traz alguns resultados para o Adulto 2 - `56`; `Private`; `10th`; `6`; `Married-civ-spouse`; `Craft-repair`; `Husband`; `White`; `Male`; `0`; `4500`; `30`; `Philippines`; `<50K` - e níveis de similaridade 0.7, 0.8 e 0.9. Não houve resultado para similaridade 0.95. Para similaridade 0.9 houve um único resultado.

Tabela 2.4: Alguns resultados da busca por similaridade na base de adultos. Adulto 1, níveis de similaridade 0.7, 0.8, 0.9

Adulto consultado
43; Self-emp-not-inc; 5th-6th; 3; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 4700; 20; United-States; <50K
Similaridade 0.7
47; Self-emp-not-inc; Some-college; 11; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 40; United-States; <50K
46; Private; Bachelors; 14; Married-civ-spouse; Transport-moving; Husband;White; Male; 0; 3000; 20; Canada; ≥50K
46; Self-emp-not-inc; 5th-6th; 4; Never-married; Craft-repair; Own-child; White; Female; 0; 4000; 40; United-States; <50K
Similaridade 0.8
42; Self-emp-not-inc; HS-grad; 10; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; United-States; <50K
43; Private; HS-grad; 10; Married-civ-spouse; Transport-moving; Husband; Black; Male; 0; 2500; 20; United-States; <50K
64; Self-emp-not-inc; 5th-6th; 4; Married-civ-spouse; ?; Wife; White; Female; 0; 4000; 20; United-States; <50K
Similaridade 0.9
56; Self-emp-not-inc; 5th-6th; 4; Married-civ-spouse; Craft-repair; Husband; White; Male; 1000; 0; 30; United-States; <50K

A Tabela 2.6 traz resultados para o Adulto 3 - 50; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; ≥50K. Nesse caso, houve resultados para todos os níveis de similaridade utilizados.

A Tabela 2.7 traz resultados para o Adulto 4 - 45; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K. Nesse caso, houve resultados para todos os níveis de similaridade utilizados.

Finalmente, a Tabela 2.8 traz resultados para o Adulto 5 - 30; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 30; United-States; ≥50K. Nesse caso, houve resultados para todos os níveis de similaridade utilizados.

A análise desses resultados é bastante subjetiva e dependente de interpretação. Entretanto, eles se mostram coerentes e é perceptível o aumento da semelhança entre os resultados e a referência em função do aumento do nível de similaridade requisitado na consulta. Dessa forma, pode-se afirmar que qualitativamente os resultados são bons e comprovam a eficácia da solução baseada na similaridade de Hamming entre os identificadores dos vetores.

2.5 Discussões Finais

Apesar da simplicidade da proposta, e da boa qualidade dos resultados obtidos, há um ponto na metodologia adotada que merece consideração por parte dos desenvolvedores que optarem pela solução. O problema refere-se ao fato de que todos os valores originais do vetor de adultos

Tabela 2.5: Alguns resultados da busca por similaridade na base de adultos. Adulto 2, nível de similaridade 0.7, 0.8, 0.9

Adulto consultado
56; Private; 10th; 6; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 4500; 30; Philippines; <50K
Similaridade 0.7
56; Private; 7th-8th; 5; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; Ecuador; <50K
31; Private; HS-grad; 10; Married-civ-spouse; Craft-repair; Husband; White; Male; 1500; 0; 30; United-States; <50K
68; Private; HS-grad; 10; Married-civ-spouse; Machine-op-inspct; Husband; White; Male; 0; 0; 40; United-States; <50K
Similaridade 0.8
66; Private; 10th; 6; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; United-States; <50K
50; Private; Bachelors; 14; Married-civ-spouse; Adm-clerical; Husband; Asian-Pac-Islander; Male; 0; 0; 30; Philippines; <50K
50; Private; Assoc-acdm; 13; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 3000; 30; United-States; >50K
Similaridade 0.9
46; Private; 10th; 6; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; ?; <50K

foram transformados em atributos do vetor modificado, que correspondem a dimensões desse novo vetor. Dessa forma os atributos do vetor original que possuem uma maior quantidade de valores estarão representados no vetor modificado em um número maior de dimensões fazendo com que esses atributos acabem, naturalmente, tendo um peso maior na definição de similaridade entre os vetores modificados.

Essa deficiência pode ser contornada atribuindo-se pesos diferentes aos atributos no momento da normalização de forma a compensar as diferenças geradas na expansão do vetor. Essa solução também poderia permitir aos usuários que definam pesos para determinados atributos em função da modelagem requerida para a sua aplicação. Entretanto, essas possibilidades não foram exploradas nesta tese visto que não é objetivo conduzir a discussão para o campo da modelagem de objetos através de vetores.

Um outro aspecto que merece atenção na metodologia proposta refere-se ao fato de que o modelo de transformação adotado não permite a realização de buscas por intervalo. Não há suporte para buscas do tipo "adultos com idades entre [18..25] anos". Entretanto, o modelo adotado facilita a busca por conjuntos de dados, como por exemplo, consultas do tipo "Adultos que nasceram em `Portugal`, `Ireland` ou `France`". Neste caso pode-se marcar todos esses três atributos no vetor mas, cabe ressaltar que o resultado dessa consulta poderá não ser exclusivo a essas três nacionalidades, visto que outros atributos poderão ser suficientes para provocar o nível de similaridade requerido.

Uma base de dados muito grande, típica do cenário de *Big Data* evocado na Introdução

Tabela 2.6: Alguns resultados da busca por similaridade na base de adultos. Adulto 3, nível de similaridade 0.7, 0.8, 0.9 e 0.95

Adulto consultado
50; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; ≥50K
Similaridade 0.7
56; Private; HS-grad; 10; Married-civ-spouse; Exec-managerial; Husband; White; Male; 0; 0; 40; United-States; ≥50K
42; Self-emp-inc; Bachelors; 14; Married-civ-spouse; Exec-managerial; Husband; White; Male; 1500; 0; 40; United-States; ≥50K
38; Private; Prof-school; 15; Never-married; Prof-specialty; Own-child; White; Female; 0; 0; 36; United-States; ≥50K
Similaridade 0.8
46; Self-emp-inc; HS-grad; 10; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; United-States;≥50K
44; State-gov; Bachelors; 14; Married-civ-spouse; Exec-managerial; Husband; White; Male; 0; 0; 40; ?; ≥50K
44; State-gov; Masters; 16; Married-civ-spouse; Prof-specialty ; Husband; White; Male; 0; 0; 36; United-States;≥50K
Similaridade 0.9
61; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; Greece; ≥50K
41; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; Japan; ≥50K
31; Self-emp-inc; Masters; 16; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; ≥50K
Similaridade 0.95
60; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; ≥50K
46; Self-emp-inc; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 36; United-States; ≥50K

desta tese, não pode ser implementada de maneira centralizada. Uma maneira interessante de viabilizar o processamento de grandes volumes de dados é a distribuição dos mesmos em vários servidores com a utilização de soluções de paralelismo. Nos próximos capítulos são apresentadas duas propostas de soluções de infraestrutura voltadas para a distribuição de grandes massas de dados e que exploram as características da similaridade de Hamming apresentadas neste capítulo.

Tabela 2.7: Alguns resultados da busca por similaridade na base de adultos. Adulto 4, nível de similaridade 0.7, 0.8, 0.9 e 0.95

Adulto consultado
45; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
Similaridade 0.7
33; Private; Bachelors; 14; Married-civ-spouse; Adm-clerical; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
44; Private; Doctorate; 18; Married-spouse-absent; Prof-specialty; Not-in-family; Asian-Pac-Islander; Male; 0; 0; 40; China; <50K
66; Private; Masters; 16; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; <50K
Similaridade 0.8
33; Private; Bachelors; 14; Married-civ-spouse; Tech-support; Husband; White; Male; 0; 0; 36; United-States; <50K
45; Private; Masters; 16; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; England; $\geq$50K
40; Private; HS-grad; 10; Married-civ-spouse; Sales; Husband; White; Male; 0; 0; 36; United-States; <50K
Similaridade 0.9
42; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
31; Private; Masters; 16; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
31; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; France; $\geq$50K
Similaridade 0.95
55; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
60; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K
75; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; $\geq$50K

Tabela 2.8: Alguns resultados da busca por similaridade na base de adultos. Adulto 5, nível de similaridade 0.7, 0.8, 0.9 e 0.95

Adulto consultado
30; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 30; United-States; ≥50K
Similaridade 0.7
58; Self-emp-not-inc; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K
37; Private; Bachelors; 10; Married-civ-spouse; Exec-managerial; Husband; White; Male; 0; 0; 36; United-States; ≥50K
38; Private; Prof-school; 15; Never-married; Adm-clerical; Own-child; White; Female; 0; 0; 30; United-States; ≥50K
Similaridade 0.8
32; Private; Doctorate; 18; Never-married; Prof-specialty; Own-child; White; Male; 0; 0; 40; United-States; <50K
38; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; Asian-Pac-Islander; Male; 0; 0; 40; Taiwan; ≥50K
36; Self-emp-inc; Masters; 16; Married-civ-spouse; Craft-repair; Husband; White; Male; 0; 0; 30; United-States; <50K
Similaridade 0.9
44; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K
48; Private; Some-college; 11; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K
39; Private; Doctorate; 18; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; ?; ≥50K
Similaridade 0.95
42; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K
45; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K
38; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 40; United-States; ≥50K

Capítulo 3

Hamming DHT

Neste capítulo será apresentada a Hamming DHT, uma rede "par a par" sobreposta (*P2P overlay*) voltada para a distribuição de conteúdo e cujo desenvolvimento privilegia a busca por similaridade. A principal motivação desta solução é o fato de que as redes P2P são eficientes na distribuição de conteúdo (Girdzijauskas 2009) e capazes de suportar o armazenamento de grandes volumes de dados, tornando-se uma alternativa interessante no cenário de *Big Data*.

O capítulo traz uma breve história das redes P2P e suas classificações segundo a literatura focando, especialmente, no conceito de DHT (*Distributed Hash Table*), base do desenvolvimento da Hamming DHT, uma das principais contribuições desta tese. Em seguida, a Hamming DHT será apresentada e terá seu funcionamento detalhado. O objetivo nesse caso é mostrar a adequação da solução proposta à busca por similaridade de conteúdos (objetos) representados por vetores cujos índices foram gerados através da função RHH. Para comprovar a hipótese levantada na proposta da Hamming DHT e o cumprimento dos objetivos estabelecidos, o fim do capítulo traz os resultados de algumas avaliações realizadas através da implementação de um protótipo para a Hamming DHT.

3.1 Redes P2P e DHTs

Embora o termo "par-a-par" (P2P, ou "peer-to-peer") seja relativamente novo na história da Internet, o seu conceito é mais antigo e vem desde o princípio da grande rede mundial, suportando serviços básicos para o funcionamento dessa rede até os dias atuais. Como exemplo podemos citar o serviço de nomes (DNS, ou *Domain Name System*) e o serviço de correio eletrônico (SMTP, ou *Simple Mail Transfer Protocol*). Ambos são descentralizados por natureza, com cada participante colaborando com os demais para garantir o correto funcionamento dos serviços.

O início da era de compartilhamento de arquivos aconteceu por volta dos anos 2000 e 2001 com o surgimento do Napster (Fanning, Fanning & Parker 2013) e, posteriormente, com a rede Gnutella (Frankel, Rubinacci & Pepper 2013). Enquanto a primeira possuía uma arquitetura centralizada, com a existência de um ponto crítico de falha, o que causou o seu declínio, a segunda seguiu uma arquitetura robusta e descentralizada porém ainda desestruturada e baseada em mecanismos de inundação de mensagens na rede. Essa arquitetura descentralizada serviu de

base para o desenvolvimento de quase todas as propostas de soluções futuras.

A transformação fundamental no cenário das redes P2P ocorreu com o surgimento das redes sobrepostas (*overlay*) estruturadas, baseadas em algoritmos de roteamento mais eficientes do que a técnica da inundação e possuindo topologias não-arbitrárias, tais como anel (Stoica et al. 2003), *torus* (Ratnasamy, Francis, Handley, Karp & Schenker 2001), hipercubos (Schlosser, Sintek, Decker & Nejdl 2003), malha (Rowstron & Druschel 2001) ou grafos *de-Bruijn* (Kaashoek & Karger 2003). Dentre as redes P2P estruturadas destacam-se as *Distributed Hash Tables* (DHT), que provêem um serviço de busca e recuperação eficiente de conteúdos baseado no modelo chave-valor (*key-value*) onde se faz uso de alguma função de *hash* para geração das chaves (*key*) de indexação para os conteúdos. Os valores (*values*), que representam os conteúdos, tais como música, vídeo, imagens, textos e arquivos, são indexados e distribuídos na DHT e essa possui um algoritmo eficiente para atribuir as responsabilidades de cada participante da rede no armazenamento dos conteúdos. Em geral, as DHTs apresentam duas primitivas básicas de serviço: *put*(k,v) e *get*(k). A primeira refere-se ao armazenamento de um valor v associado a uma chave k e a segunda refere-se à recuperação do conteúdo indexado por k a partir de algum nó participante da DHT. Há ainda a possibilidade dos nós da DHT não armazenarem os conteúdos, mas sim, referências aos mesmos, como por exemplo uma lista de participantes da rede que disponibilizam o conteúdo k para *download*, como uma espécie de catálogo.

Dentre as diversas propostas de DHT destacam-se o Chord (Stoica et al. 2003), Pastry (Rowstron & Druschel 2001), Tapestry (Zhao, Huang, Stribling, Rhea, Joseph & Kubiatowicz 2004) e CAN (Ratnasamy et al. 2001). Assim como acontece nas tabelas *hash* tradicionais, todos os participantes da DHT compartilham uma mesma função de *hash*. Entretanto, em uma DHT, os participantes da rede dividem o espaço de endereçamento de modo cooperativo e, consequentemente, a responsabilidade pelo armazenamento dos valores associados às chaves de indexação, os identificadores. Além disso, os nós participantes da rede mantêm uma tabela de roteamento capaz de garantir o acesso global a todo o espaço de endereçamento da DHT de modo eficiente (normalmente com custo logarítmico).

Formalmente podemos descrever uma rede P2P como sendo um conjunto de nós $\mathcal{P}$ que provêem acesso a um conjunto de conteúdos $\mathcal{C}$ através de uma função que mapeie $\mathcal{P}$ e $\mathcal{C}$ em um espaço virtual de identificadores $\mathcal{I}$ utilizando-se de duas funções:

$\mathcal{F}_{\mathcal{P}} : \mathcal{P} \rightarrow \mathcal{I}$, e
$\mathcal{F}_{\mathcal{C}} : \mathcal{C} \rightarrow \mathcal{I}$.

Essas funções permitem o estabelecimento de relações de distância e pertinência entre conteúdos e nós através de métricas no espaço de endereçamento $\mathcal{I}$.

Para permitir o acesso aos nós, e consequentemente aos conteúdos compartilhados na rede, torna-se necessário embutir um grafo de conectividade a esse espaço de identificadores. Esse grafo auxilia o roteamento facilitando a recuperação de conteúdos associados aos identificadores.

Um projeto de redes P2P deve considerar os seguintes aspectos:

- a escolha do espaço de identificadores $\mathcal{I}$;
- a escolha das funções de mapeamento $\mathcal{F}_{\mathcal{P}}$ e $\mathcal{F}_{\mathcal{C}}$;

- a pertinência de conteúdos aos nós;
- o grafo de conectividade dos nós embutido no espaço de identificadores;
- a estratégia de roteamento na rede;
- a manutenção da consistência da rede em função da entrada e saída de nós.

As opções realizadas durante o desenvolvimento da Hamming DHT serão detalhadas e explicadas nas subseções a seguir:

3.1.1 Espaço de endereçamento

Uma característica fundamental na escolha do espaço de endereçamento é a existência de alguma métrica de distância d: $\mathcal{I}$ x $\mathcal{I} \rightarrow \mathbb{R}$, onde $\mathbb{R}$ denota o conjunto dos números reais. Essa métrica de distância d deve satisfazer as seguintes propriedades:

$\forall x, y \in \mathcal{I} : d(x, y) \geq 0$
$\forall x \in \mathcal{I} : d(x, x) = 0$
$\forall x, y \in \mathcal{I} : d(x, y) = 0 \rightarrow x = y$
$\forall x, y \in \mathcal{I} : d(x, y) = d(y, x)$
$\forall x, y, z \in \mathcal{I} : d(x, z) \leq d(x, y) + d(y, z)$

A escolha adequada do espaço de identificadores de nós e conteúdos é uma questão relevante pois em função dessa escolha dependerão fatores como a escalabilidade e o custo do roteamento na DHT.

3.1.2 Funções de mapeamento

A função de mapeamento $\mathcal{F}_{\mathcal{P}}$ associa um nó a um ponto no espaço $\mathcal{I}$. Na maioria das redes P2P escolhe-se funções de mapeamento onde esse ponto é único, tal que:

$\forall p,q \in \mathcal{R} : p \neq q \rightarrow \mathcal{F}_{\mathcal{P}}(p) \neq \mathcal{F}_{\mathcal{P}}(q)$

Essa mesma condição pode ser aplicada à função $\mathcal{F}_{\mathcal{C}}$, associando um conteúdo na rede a um ponto no espaço de identificadores. Em geral não existe nenhuma relação de similaridade entre os identificadores gerados e o conteúdo associados a eles e privilegia-se uma distribuição homogênea dos conteúdos no espaço de endereçamento. Funções de *hash* como MD5 (*Message-Digest Algorithm 5*) ou SHA-1 (*Secure Hash Algorithm 1*) possuem distribuição homogênea e são geralmente usadas nesses casos. Em uma distribuição homogênea os identificadores gerados tendem a se espalhar de maneira uniforme no espaço de endereçamento.

O relaxamento dessa condição representa a probabilidade de conteúdos similares ocuparem o mesmo ponto no espaço de identificadores. A probabilidade de um nó ou conteúdo ser mapeado em um determinado ponto ou região do espaço virtual $\mathcal{I}$ também é uma característica importante na escolha das funções $\mathcal{F}_{\mathcal{P}}$ e $\mathcal{F}_{\mathcal{C}}$. Funções LSH possuem distribuições heterogêneas,

onde os identificadores são mapeados em regiões do espaço em função da relação de similaridade entre eles. Conforme visto no Capítulo 1, as funções LSH (*Locality Sensitive Hashing*) possuem a característica de manter a similaridade dos conteúdos $\mathcal{C}$ mapeados em $\mathcal{I}$. Nesse caso, a função de mapeamento $\mathcal{F}_\mathcal{C}$ deve satisfazer às seguintes relações:

$\forall\ c_1,c_2 \in \mathcal{C} : d(\mathcal{F}_\mathcal{C}(c_1),\mathcal{F}_\mathcal{C}(c_2)) \propto 1/sim(c_1,c_2)$
$\forall\ c_1,c_2 \in \mathcal{C} : \mathbf{Pr}[\mathcal{F}_\mathcal{C}(c_1) = \mathcal{F}_\mathcal{C}(c_2)] = sim(c_1,c_2)$

A partir dessas relações pode-se afirmar que a distância entre dois conteúdos quaisquer é inversamente proporcional à similaridade entre eles, ou seja, quanto mais similares, mais próximos no espaço de endereçamento eles se encontram, e vice-versa.

3.1.3 Roteamento em uma DHT

O serviço básico provido por uma DHT é o encaminhamento das solicitações de recuperação de conteúdos representados por seus identificadores k a partir de um nó p. A estratégia de roteamento adotada na rede influencia na distância entre nós e identificadores representada pelo número de saltos necessários para o encaminhamento da requisição desde p até o nó p' responsável pelo armazenamento da chave k na DHT.

Uma função de roteamento $\mathcal{F}_\mathcal{R}$ pode ser definida como uma função não determinística:

$\mathcal{F}_\mathcal{R} : \mathcal{P}$ x $\mathcal{I} \rightarrow \mathcal{P}$

que seleciona, no conjunto $\mathcal{N}_\mathcal{P}$ de nós vizinhos a p, um outro nó $p' \in \mathcal{N}_\mathcal{P}$ para encaminhar a solicitação. O nó escolhido para o encaminhamento, p', deverá satisfazer a seguinte relação:

$d(k,\mathcal{F}_\mathcal{P}(p')) < d(k,\mathcal{F}_\mathcal{P}(p))$

3.2 Apresentação da Hamming DHT

A maioria das DHTs não foi projetada especificamente para suportar a busca por similaridade. No Capítulo 5 serão apresentados trabalhos relacionados e ficará evidente o posicionamento da Hamming DHT no contexto da busca por similaridade em DHTs.

A Hamming DHT diferencia-se das demais propostas por ser projetada especialmente para suportar a busca por similaridade de conteúdos representados por vetores cujos identificadores foram gerados através da função de *hash* RHH. As duas principais características do projeto da Hamming DHT são:

- Espaço de identificadores único, organizado sob a forma de anel virtual, compartilhado entre nós e conteúdos e ordenados segundo a sequência do código de Gray, onde identificadores sucessivos apresentam distância de Hamming igual a 1 (um) e, portanto, representam conteúdos muito similares;

- Criação da tabela de roteamento em função da distância de Hamming entre os nós, de tal forma que a função de roteamento adotada reflita em seus saltos a distribuição dos identificadores similares na distância de Hamming no espaço de endereçamento mapeado no anel.

A proposta da Hamming DHT é reduzir a distância, medida pelo número de saltos realizados pelas funções de roteamento, na busca por similaridade. Como consequência dessa redução de distância, a cobertura (*recall*) das buscas será aumentada, com menos esforço, permitindo assim a criação de serviços mais eficientes de busca por similaridade em ambientes distribuídos tipo DHT. Conforme já definido anteriormente, em sistemas de Recuperação de Informação define-se a cobertura como sendo a relação entre os conteúdos relevantes recuperados com êxito pelo serviço de busca em relação ao total dos conteúdos relevantes existentes no sistema (Berry et al. 1999). Essa mesma definição será usada nesta tese em um serviço de busca por similaridade.

3.2.1 Organização dos identificadores

Na Hamming DHT tanto os conteúdos quanto os nós compartilham um mesmo espaço de endereçamento formado por identificadores de m *bits* organizados sob a forma de um anel virtual cuja sequência obedece ao código de Gray. Essa é uma primeira característica que diferencia a Hamming DHT da maioria das DHTs que adotam a topologia de anel virtual para organizar os seus identificadores. Conforme já dito anteriormente, na sequência do código de Gray identificadores sucessivos apresentam distância de Hamming igual a 1 (um) e, portanto, no contexto desta tese, representam, probabilisticamente, conteúdos muito similares.

A função de mapeamento dos nós no espaço de endereçamento, $\mathcal{F}_{\mathcal{P}}$, usa funções de *hash* tradicionais, tais como MD5 (*Message-Digest Algorithm 5*) ou SHA-1 (*Secure Hash Algorithm 1*). O objetivo dessa escolha é espalhar os nós de forma homogênea no espaço de endereçamento e garantir a unicidade, com alta probabilidade, dos identificadores de nós na rede. É importante ressaltar aqui que o objetivo principal da proposta da Hamming DHT é promover a similaridade de Hamming entre conteúdos, e não entre nós. A função dos nós é armazenar os conteúdos e, principalmente, manter a integridade da DHT.

Por outro lado, a função de mapeamento dos conteúdos, representados pelos seus vetores, e o espaço de endereçamento, $\mathcal{F}_{\mathcal{C}}$, usa a função de *hash* RHH, capaz de representar a similaridade entre os vetores de conteúdo na distância de Hamming entre os seus identificadores. Nesse caso a distribuição de conteúdos não é homogênea e admite-se que conteúdos muito similares possuam o mesmo identificador na rede - com probabilidade igual à similaridade entre eles, medida pelo cosseno entre seus vetores - e concentrem-se em determinadas regiões do anel virtual.

A Figura 3.1 mostra um exemplo de um anel virtual de uma Hamming DHT com $m = 5$ bits; quatro nós distribuídos aleatoriamente com identificadores iguais a 3 (00011_2), 13 (01101_2), 30 (11110_2) e 22 (10110_2); e três conteúdos quaisquer com identificadores 0 (00000_2), 24 (11000_2) e 31 (11111_2). Nesse ponto cabe destacar que a sequência de ordenação do código de Gray é diferente da ordenação binária natural. Observa-se na figura que, de acordo com o código de Gray, $3 < 13 < 30 < 22$.

Um anel virtual com m *bits* possui $2^m - 1$ identificadores que variam de acordo com a sequência de ordenação adotada para o anel. No caso da ordenação binária natural os identificadores

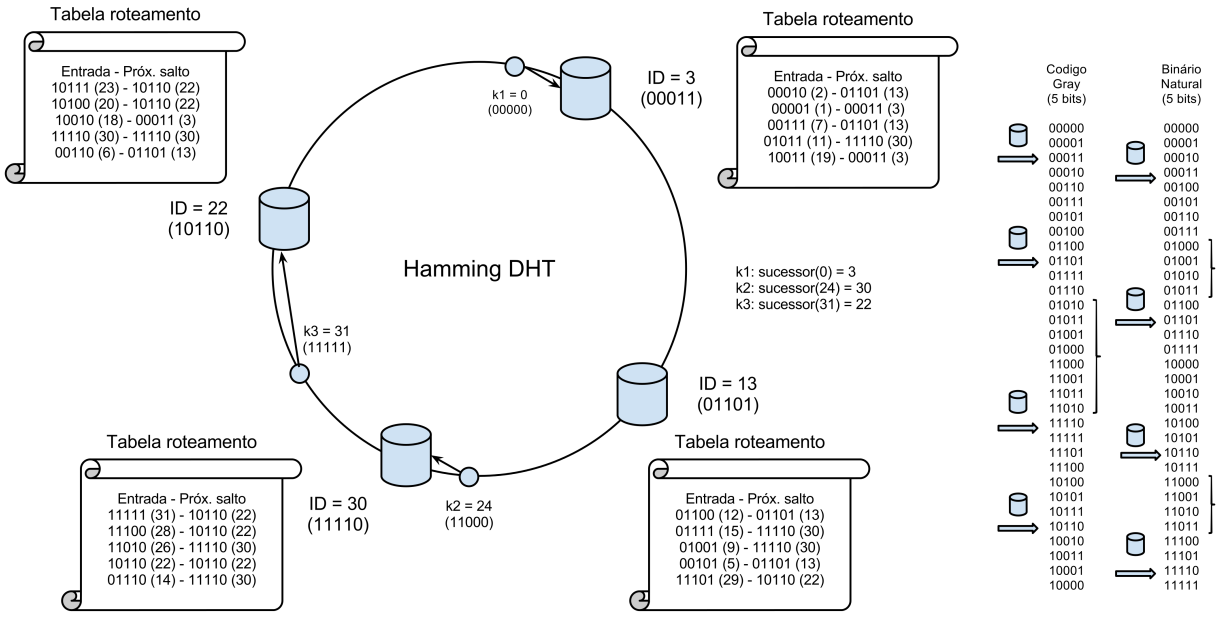


Figura 3.1: Organização de uma Hamming DHT com $m = 5$ bits.

variam no intervalo $[0..2^m - 1]$ sendo que 0 é o sucessor do identificador $2^m - 1$. No caso da ordenação segundo o código de Gray binário refletido o 0 é o sucessor do $(2^m - 1)$-ésimo identificador, que para $m = 5$ *bits* tem valor igual a 16 (10000_2). Em ambos os casos, a existência de um sucessor para o identificador 0 é o que dá a característica de anel virtual ao espaço de endereçamento virtual.

3.2.2 Atribuição de conteúdos aos nós na Hamming DHT

Em uma DHT os nós participantes da rede dividem de forma colaborativa a responsabilidade pela gerência do espaço de endereçamento. Desta forma diz-se que cada nó é responsável por parte desse espaço. Uma abordagem de *hash* consistente possui a caracteristica de dividir, de forma equilibrada, a responsabilidade dos nós pelo armazenamento dos identificadores de conteúdo. Em situações de entrada/saída de nós na rede, em um DHT com função de *hash* consistente determina-se, facilmente, quais serão os identificadores que deverão ser atribuídos aos nós que acabaram de entrar e quais eram os identificadores atribuídos aos nós que acabaram de sair da rede, sem afetar os demais nós presentes na DHT.

Nas DHTs tradicionais, o uso de funções de *hash* homogêneas, tais como MD5 e SHA-1, leva, com alta probabilidade, ao balanceamento da rede. Nesse caso os nós tendem a se espalhar de forma homogênea ocupando todo o espaço de endereçamento. A mesma consideração é feita para os conteúdos indexados através dessas funções. Em (Stoica et al. 2003) há a prova formal de que o uso destas funções torna, com alta probabilidade, a DHT consistente. Na Hamming DHT o uso de funções de *hash* homogêneas para geração dos identificadores dos nós provoca o espalhamento dos mesmos na DHT, de tal forma que eles fiquem responsáveis por aproximadamente a mesma quantidade de identificadores. Entretanto, o uso de funções de *hash* não homogêneas na geração de identificadores de conteúdo faz com que, embora os nós sejam

responsáveis por aproximadamente a mesma quantidade de identificadores, isso não implica que eles armazenarão aproximadamente a mesma quantidade de conteúdo, uma vez que os conteúdos são agrupados por similaridade e podem se concentrar em determinadas regiões do anel.

Em geral, nas DHTs baseadas em anel virtual, todo nó é responsável pelo espaço compreendido entre o identificador do nó que o antecede e o seu próprio identificador, com os identificadores ordenados de modo crescente no sentido horário do anel. No exemplo da Figura 3.1 diz-se que o nó 13 (01101_2) é responsável por todos os identificadores compreendidos entre o nó 3 (00011_2), seu antecessor, e ele mesmo, nó 13 (01101_2). Formalmente, se p é o identificador de um nó, e *antecessor(p)* é uma função que retorna o antecessor do nó p na DHT, diz-se que p é responsável por todos os identificadores k no intervalo $antecessor(p) < k \leq p$. A partir deste ponto consideraremos sempre que p é o identificador do nó p, ao invés de usarmos a notação formal $\mathcal{F}_{\mathcal{P}}(p)$.

O antecessor de um nó p em uma DHT baseada em anel virtual no sentido horário é o nó *antecessor(p)* que possui o maior identificador dentre todos os nós da DHT que possuem identificadores menores do que p. Por outro lado, o sucessor de um nó p em uma DHT baseada em anel virtual no sentido horário é o nó *sucessor(p)* que possui o menor identificador dentre todos os nós da DHT que possuem identificadores maiores do que p. Referente à mesma figura, a Tabela 3.1 traz os sucessores e os identificadores de todos os nós da Hamming DHT usada como exemplo:

Tabela 3.1: Antecessores e Sucessores dos nós na Hamming DHT da Figura 3.1.

Nó	Sucessor	Antecessor
3 (00011_2)	13 (01101_2)	22 (10110_2)
13 (01101_2)	30 (11110_2)	3 (00011_2)
30 (11110_2)	22 (10110_2)	13 (01101_2)
22 (10110_2)	3 (00011_2)	30 (11110_2)

A partir da Tabela 3.1 pode-se afirmar que em uma DHT cuja ordenação segue a sequência dos números naturais, o nó 30 seria sucessor do nó 22, que seria sucessor do nó 13. O nó 13, por sua vez, seria antecessor do nó 22, que seria antecessor do nó 30, que seria o antecessor do nó 3.

A simplicidade ao se realizar o mapeamento entre nós e conteúdos, inspirada no Chord (Stoica et al. 2003), é uma grande vantagem da Hamming DHT. Em se tratando da implementação, o fato do código de Gray possuir algoritmos simples de conversão para a sequência natural (Algoritmo 2), e vice-versa (Algoritmo 1), facilita a implementação desse mapeamento. Chamaremos o Algoritmo 2 de *grayToBinary* e o Algoritmo 1 de *binaryToGray*.

Na Hamming DHT, para determinar se um identificador de conteúdo k é de responsabilidade do nó p, cujo antecessor é *ant*, usa-se o Algoritmo 3, apresentado a seguir.

Usando a Figura 3.1, os nós 30 (11110_2) e 22 (10110_2) e o conteúdo 31 (11111_2) como exemplo, tem-se que:

Algorithm 3: Verificação de pertinência do identificador k ao nó p.

```
Input  : Content Identificator k;
Input  : Node Identificator p, ant;
Output: Boolean result;
result ← False;
ant ← antecessor(p);
if (grayToBinary(p) == grayToBinary(k)) then
 |  result ← True;
end
if (grayToBinary(p) > grayToBinary(ant)) then
 |  if (grayToBinary(ant) < grayToBinary(k) ≤ grayToBinary(p)) then
 |  |  result ← True;
 |  end
end
if (grayToBinary(p) < grayToBinary(ant)) then
 |  if ((grayToBinary(k) > grayToBinary(ant)) || (grayToBinary(k) <
 |  grayToBinary(p))) then
 |  |  result ← True;
 |  end
end
```

$grayToBinary(30) \rightarrow 20$
$grayToBinary(31) \rightarrow 21$
$grayToBinary(22) \rightarrow 27$

Dessa forma, como na sequência binária natural $20 < 21 \leq 27$, pode-se afirmar que o identificador 31 (11111_2) está entre os nós 30 (11110_2) e 22 (10110_2) na sequência de Gray, pertencendo, portanto, a um intervalo na Hamming DHT cuja responsabilidade é do nó 22 (10110_2). Baseando-se nessas funções define-se a primeira primitiva da Hamming DHT, que é a função *lookup(k)*. A função *lookup(k)* é executada de forma distribuída e recebe um identificador k como entrada e retorna o identificador do nó p responsável pelo armazenamento de k, na DHT. Em outras palavras, *lookup(k)* retorna o *sucessor(k)*. Localmente, cada nó, tendo como base o seu identificador, ao receber uma mensagem *lookup(k)* executa o Algoritmo 4:

Nesse algoritmo, cada nó local, com identificador $myId$, verifica se a chave k pertence ao intervalo sob sua responsabilidade. Em caso afirmativo, o algoritmo retorna ao chamador o seu próprio identificador $myId$. Caso contrário, o nó encaminha a mensagem *lookup(k)* para o seu sucessor no anel para que ele faça a mesma verificação através da chamada à função *forward(k)*.

É condição essencial ao funcionamento de qualquer DHT baseada em anel virtual que todos os nós participantes da rede conheçam o seu antecessor e o seu sucessor. Essa é a garantia de integridade da DHT e, dessa forma, todos os identificadores são acessíveis a partir de qualquer ponto da rede. A manutenção atualizada dos identificadores desses dois nós é função básica de todo nó da DHT e deve ser realizada através de verificações periódicas do tipo *keep-alive*. Neste caso, a Hamming DHT pode herdar do Chord (Stoica et al. 2003) o procedimento de manutenção de sucessor e antecessor, assim como todos os procedimentos de entrada e saída

Algorithm 4: Implementação da função *lookup(k)*.

Input : Content Identificator k;
Input : My Identificator $myId$;
Output: Node Identificator p;
$ant \leftarrow antecessor(p)$;
if $grayToBinary(ant) < grayToBinary(k) \leq grayToBinary(p)$ **then**
  $p \leftarrow myId$;
end
else
  forward(k)
end

de nós na DHT. Há, também, o trabalho em (Zave 2012) que propõe correções ao protocolo de manutenção do anel proposto pelo Chord e que serve de referência para futuras implementações da Hamming DHT.

3.2.3 Roteamento na Hamming DHT

Baseando-se na integridade e consistência do anel é possível procurar por qualquer identificador a partir de qualquer ponto da rede. Por exemplo, na Figura 3.1, caso um usuário procure pelo identificador $k = 24$ (11000_2) a partir do nó 3 (00011_2), o mesmo irá perceber que não é responsável por esse identificador de conteúdo e encaminhará ao seu sucessor no sentido horário do anel, o nó 13 (01101_2), que também não é responsável por k e também encaminhará ao seu sucessor, o nó 30 (11110_2). Esse, por sua vez, ao verificar o intervalo de pertinência, descobre ser o responsável por armazenar esse conteúdo (ou referências a ele) e responderá à solicitação do usuário.

É fácil notar que essa forma de roteamento é ineficiente, uma vez que ela é proporcional ao número de nós participantes da rede, com custo $N - 1$ no pior caso, onde N é o número de nós participando da rede. Isso acontece porque o encaminhamento ocorre sempre na direção do sucessor e, no pior caso, se o conteúdo cujo identificador é k está sob responsabilidade do antecessor de um nó p, a procura será encaminhada sempre no sentido horário, de sucessor em sucessor, até completar uma volta no anel e alcançar o antecessor de p passando por $N - 1$ nós.

Visando diminuir o número de saltos durante o encaminhamento de uma mensagem, reduzindo a latência na busca/recuperação de um conteúdo, em geral os nós das DHTs mantêm uma tabela de roteamento conhecida como tabela de contatos (*fingers*). As entradas dessa tabela apontam para nós mais distantes no anel possibilitando saltos maiores no espaço de endereçamento. O número de entradas na tabela de roteamento e a escolha dessas entradas é questão importante no projeto da DHT e, nesse ponto, há a outra grande diferença entre a Hamming DHT e as demais DHTs encontradas na literatura. Os nós da Hamming DHT escolhem as entradas que farão parte da sua tabela de roteamento em função da distância de Hamming entre os identificadores.

A Hamming DHT possui m entradas em sua tabela de roteamento, onde m é o comprimento, em *bits*, do espaço de endereçamento. A i-ésima entrada na tabela de roteamento (f_i) de um

nó p da Hamming DHT corresponde ao identificador gerado pela troca do i-ésimo *bit* do identificador p. Em outras palavras, f_i é um identificador que possui distância de Hamming igual a 1 (um) com p. Formalmente:

$$f_i = p \oplus 2^i \rightarrow sucessor(f_i)$$

onde $\oplus$ é a função de OU-exclusivo *bit* a *bit*.

As entradas da tabela de roteamento mapeiam o identificador f_i no identificador do nó que o sucede no anel virtual, em outras palavras, o nó responsável pelo seu armazenamento. Além disso, como se trata de uma rede overlay, nesse caso é necessário incluir o identificador do sucessor na tabela de roteamento e também o seu localizador.

Tomando-se novamente a Figura 3.1 como exemplo, nela estão exibidas as tabelas de roteamento dos 4 (quatro) nós da DHT: 3 (00011_2), 13 (01101_2), 30 (11110_2) e 22 (10110_2). Considerando-se o nó 13 (01101_2), cujas entradas e respectivos mapeamentos encontram-se indicados na Tabela 3.2, observe que a 1^a entrada da tabela do nó 13 (01101_2) corresponde à troca do *bit* 0 (menos significativo) de p, gerando o identificador f_i que é mapeado no nó que é seu sucessor no anel, ou seja, o nó responsável por f_i, ou seja, $sucessor(f_i)$.

Tabela 3.2: Tabela de Roteamento do nó 13 (01101_2) na Hamming DHT da Figura 3.1.

Entrada i	Identificador f_i	Sucessor f_i
0	12 (01100_2)	13 (01101_2)
1	15 (01111_2)	30 (11110_2)
2	9 (01001_2)	30 (11110_2)
3	5 (00101_2)	13 (01101_2)
4	29 (11101_2)	22 (10110_2)

Esse mapeamento é feito na DHT através da mensagem de *lookup*, vista anteriormente. No caso, para cada entrada f_i da tabela faz-se o $lookup(f_i)$ e adiciona-se a resposta da mensagem ao mapeamento na tabela. Lembrando que em geral, além do identificador, adiciona-se o localizador do nó na tabela de roteamento, por exemplo, o seu endereço IP. O localizador é necessário por se tratar de uma rede sobreposta (*overlay*), onde o encaminhamento de dados é feito usando-se a infraestrutura de rede disponível, baseada na arquitetura TCP/IP. É importante destacar também que as entradas f_i não representam nós, mas sim, identificadores que são mapeados para os nós que os sucedem na DHT.

Desta forma a característica da tabela de roteamento de um nó é que ela aponta para identificadores cuja distância de Hamming é 1 com o identificador de seu próprio nó. Embora não exista similaridade entre os nós da DHT, a ideia que motivou essa proposta é refletir a distribuição dos identificadores similares (segundo a distância de Hamming) no anel organizado segundo o código de Gray. A Figura 3.2 compara a distribuição dos contatos no Chord (Stoica et al. 2003) e a distribuição dos identificadores semelhantes segundo o código de Gray e as

distâncias de Hamming iguais a 1. A partir da figura percebe-se que os identificadores similares segundo a distância de Hamming se distribuem de forma distinta à organização dos contatos da tabela do roteamento do Chord (Stoica et al. 2003), tomada como exemplo. O objetivo da construção da tabela adotada pela Hamming DHT é refletir essa distribuição de identificadores similares nos contatos estabelecidos por um nó participante da rede. As avaliações realizadas na Seção 3.3 têm como objetivo mostrar que essa construção tende a privilegiar a busca por identificadores similares.

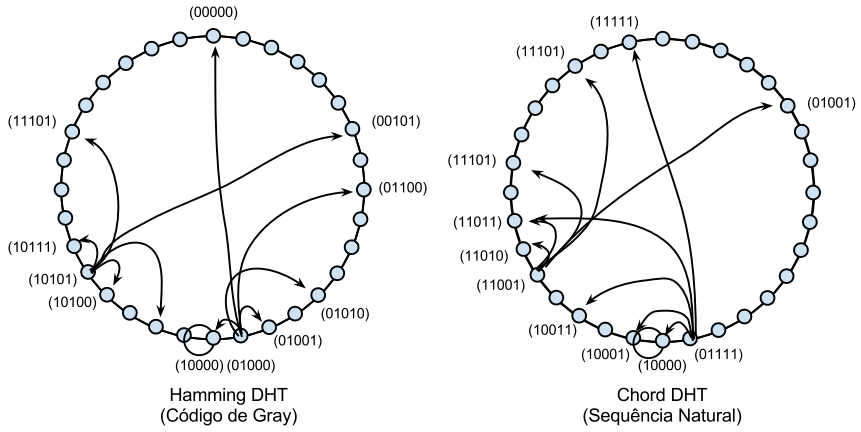


Figura 3.2: Distribuição dos contatos na Hamming DHT com código de Gray e na Chord DHT com a sequência natural. Em ambos os casos $m = 5$ bits.

Assim que essa tabela de roteamento estiver pronta, ao receber uma solicitação de consulta a um identificador k, um nó p deve, primeiramente, verificar se k pertence ao intervalo de identificadores sob a sua responsabilidade, tarefa realizada pelo Algoritmo 3. Se não estiver, p deverá encaminhar k diretamente ao nó p' presente em sua tabela de roteamento cuja entrada é o limitante inferior de f_i na sequência de Gray, ou seja, f_i é a maior entrada na tabela que é menor ou igual a k na sequência de Gray. Formalmente:

$$p' \in \mathcal{T}_p : f_i \lfloor_g k,\ f_i \rightarrow p'$$

onde $\mathcal{T}_p$ é o conjunto dos identificadores dos nós que pertencem à tabela de roteamento de p e $\lfloor_g$ é a função limitante inferior adaptada para a sequência do código de Gray.

Novamente, baseando-se na Figura 3.1 e na Tabela 3.2, considerando um usuário no nó 13 (01101_2) procurando por um conteúdo cujo identificador é 18 (10010_2), essa mensagem deve ser encaminhada para o nó 22 (10110_2), quinta entrada na sua tabela de roteamento, que corresponde ao limitante inferior do identificador 18 (10010_2) na sequência. A Figura 3.3 mostra a distribuição dos contatos do nó 13 (01101_2) e o posicionamento do conteúdo 18 (10010_2) no mesmo espaço.

Por fim, a partir da Figura 3.2, percebe-se que a agregação de conteúdos similares na distância de Hamming não concentra conteúdos em uma única região do anel virtual, como acontece com as agregações na distância Euclideana. Ao contrário, as agregações na distância de Ham-

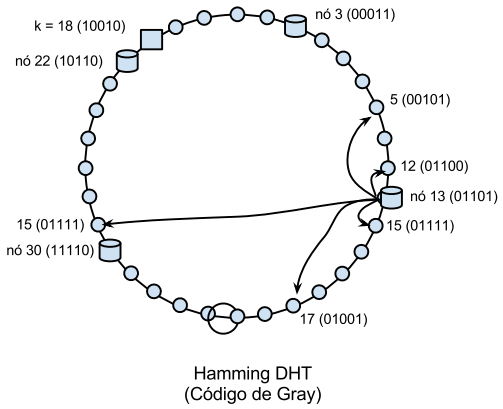


Figura 3.3: Distribuição dos contatos do nó 13 (01101_2) no espaço de endereçamento. Posicionamento do conteúdo $k = 18$ (10010_2) no mesmo espaço.

ming tendem a espalhar conteúdos em diversos pontos do anel. No final deste capítulo, Seção 3.4, faremos considerações a respeito desse problema do desbalanceamento de carga na Hamming DHT e essa característica será novamente explorada.

3.2.4 Primitiva *get*(k, sim_h)

A terceira e última característica que diferencia a Hamming DHT das outras DHTs existentes na literatura é a implementação da função $get(k, sim_h)$. Nessa função, um usuário, através de algum nó p pertencente à rede, pode fazer uma busca por um conteúdo referenciado pelo identificador k e um grau de similaridade de Hamming (sim_h). Essa consulta é então encaminhada até o nó p' responsável pelo identificador k e, localmente, p' seleciona em sua base de dados de conteúdos todos aqueles cujos identificadores possuem uma similaridade de Hamming maior ou igual à sim_h.

Além disso, explorando a característica principal da Hamming DHT que é a organização dos contatos de acordo com a distribuição dos identificadores similares no anel, p' pode encaminhar a consulta a todos os seus vizinhos (aqueles nós presentes na sua tabela de roteamento), de forma a buscar os demais conteúdos similares que estão espalhados em outras regiões do anel. Com isso, procura-se aumentar a cobertura das buscas. Esse procedimento de "encaminhar para os vizinhos" pode ser repetido pelos "vizinhos dos vizinhos" aumentando ainda mais a cobertura da busca com a contrapartida do aumento do número de mensagens na rede.

3.3 Avaliação da Hamming DHT

Para avaliar a proposta da Hamming DHT foi desenvolvido um simulador usando a linguagem de programação Java. O simulador desenvolvido possui as seguintes habilidades:

- Criar N nós virtuais em um espaço de endereçamento de comprimento m, em bits. Os identificadores dos nós são criados aleatoriamente e distribuídos de maneira uniforme;
- Fornecer as primitivas $put(k, v)$ e $get(k, sim_h)$ para realização das buscas por similaridade;

- Implementar a primitiva de *lookup*(k), permitindo que cada nó da DHT possa construir a sua própria tabela de roteamento;
- Contabilizar o número de saltos em cada *lookup*(k) ou $get(k, sim_h)$ realizado de um nó origem até o nó destino;
- Contabilizar o número de itens de conteúdo que são recuperados em cada nó após a realização de uma busca por similaridade.

Considerando que o objetivo dessa tese é mostrar que a construção da Hamming DHT favorece a realização da busca por similaridade de conteúdos usando a similaridade de Hamming através da redução da distância entre conteúdos similares e aumento da cobertura das buscas, optou-se por não implementar uma DHT completamente funcional. Dessa forma o simulador não suporta o comportamento dinâmico de uma DHT. Entretanto, vale ressaltar que essa é uma limitação do simulador e não da Hamming DHT, visto que a mesma possui um espaço de endereçamento baseado em anel virtual, *hash* consistente, funções de atribuição de conteúdos aos nós baseadas na existência do nó sucessor, crescimento dos identificadores no sentido horário do anel e tabela de roteamento com o mesmo número de entradas que o Chord (Stoica et al. 2003). Dessa forma todos os algoritmos que suportam o comportamento dinâmico do Chord (Stoica et al. 2003) podem ser diretamente adaptados à Hamming DHT.

Conforme apresentado no Capítulo 2 os testes da Hamming DHT foram realizados usando a base de dados de adultos. Os testes têm como objetivo:

1. Avaliar a distribuição de distâncias, medidas pelo número de saltos entre nós da DHT entre conteúdos similares;
2. Medir a cobertura das buscas por similaridade em função da profundidade no encaminhamento das consultas.

Para atingir esses objetivos o seguinte procedimento foi realizado: dado um conteúdo de referência k e um nível de similaridade de Hamming sim_h, avaliar a distribuição de frequência das distâncias entre os conteúdos similares a k e o nó de referência p', responsável pelo armazenamento de k. Em outras palavras, a partir de qualquer ponto do anel virtual o usuário de um nó p pode gerar uma consulta $get(k, sim_h)$ que será encaminhada até o nó p' responsável pelo armazenamento de k. Em p', todos os conteúdos indexados que possuíam similaridade de Hamming maior ou igual a sim_h (nível de similaridade fornecido na consulta) foram selecionados para a resposta. Em paralelo, encaminhou-se a consulta para todos os nós p'' presentes na tabela de roteamento de p', aumentando-se a profundidade da consulta para 1 (um). Cada nó p'' pertencente à tabela de roteamento de p' se encarrega de selecionar os conteúdos similares para a resposta de acordo com sim_h e, em paralelo, encaminhar aos nós p''' presentes em suas tabelas de roteamento, aumentando-se nesse caso a profundidade para 2 (dois). Nesse cenário de teste esse procedimento foi repetido até que todos os conteúdos similares a k fossem recuperados de tal forma a permitir a avaliação da cobertura das buscas. É importante destacar aqui que nesse cenário controlado de testes, a base de dados de adultos é previamente conhecida e, por isso, sabia-se com antecedência quais são os adultos similares e quantos eles são.

Em um cenário real a profundidade deve ser um parâmetro configurável. Os resultados mostrarão que a Hamming DHT é capaz de conseguir um bom resultado de cobertura em profundidades reduzidas.

Em todos os testes os principais parâmetros utilizados foram os seguintes:

- Identificadores de 128 *bits* gerados pela função RHH;
- DHTs com 1000 e 10000 nós;
- Níveis de similaridade iguais ou superiores a 0.7, 0.8, 0.9 e 0.95;
- Resultados comparados com o Chord;
- Cada adulto presente na base de dados Adult (Frank & Asuncion 2010) foi usado como referência nas consultas realizadas na mesma base de dados, composta por 48842 adultos, gerando um total de 48842^2 consultas.

3.3.1 Distribuição de frequência da distância (em saltos) entre conteúdos similares

A Figura 3.4 apresenta os resultados obtidos na avaliação da distribuição de frequência das distâncias, medidas em saltos, entre conteúdos similares. Os valores representam a média obtida após as simulações de 10 redes distintas com 1000 nós cujos identificadores foram gerados aleatoriamente.

Conforme dito anteriormente, os resultados comparam Chord (Stoica et al. 2003) e Hamming DHT e representam o quão distantes estão, em média, os conteúdos similares em ambas as DHTs.

A partir dos resultados apresentados na Figura 3.4 pode-se perceber que a Hamming DHT apresenta uma quantidade maior de conteúdos similares concentrados a menores distâncias do nó de referência, o nó p'. Por exemplo, para similaridade 0.9 (Figura 3.4(c)) cerca de 18% dos conteúdos similares encontram-se a 1 (um) salto de distância do nó p', e outros 12% encontram-se à distância 2 na Hamming DHT. Ainda na mesma figura pode-se constatar que nenhum conteúdo similar estará a uma distância maior do que 4 na Hamming DHT. Os mesmos resultados mostram que cerca de 8% e 7% dos conteúdos estão a 6 e 7 saltos de distâncias do nó p' na Chord DHT. Além disso há conteúdos a 9 saltos de distância no Chord (Stoica et al. 2003).

Os mesmos testes foram realizados em DHTs maiores, com 10000 nós. A Figura 3.5 traz esses resultados. Os valores também representam a média obtida em 10 simulações com redes distintas, cujos nós foram gerados aleatoriamente. Em todos os casos foi avaliado o Intervalo de Confiança estatístico (IC) das medidas, com valor de 95%. Durante o processo de avaliação foi observado que as 10 simulações eram suficientes para que as médias fossem confiáveis. Os resultados do cálculo dos IC não são visíveis na escala dos gráficos apresentados.

Os resultados da Figura 3.5 mostram que a mesma característica é apresentada com o aumento do número de nós na DHT. Assim como no resultado obtido anteriormente, para 10000 nós a Hamming DHT mantém a característica de agregar os conteúdos similares a uma menor distância na rede, quando comparada com uma DHT tradicional como o Chord (Stoica

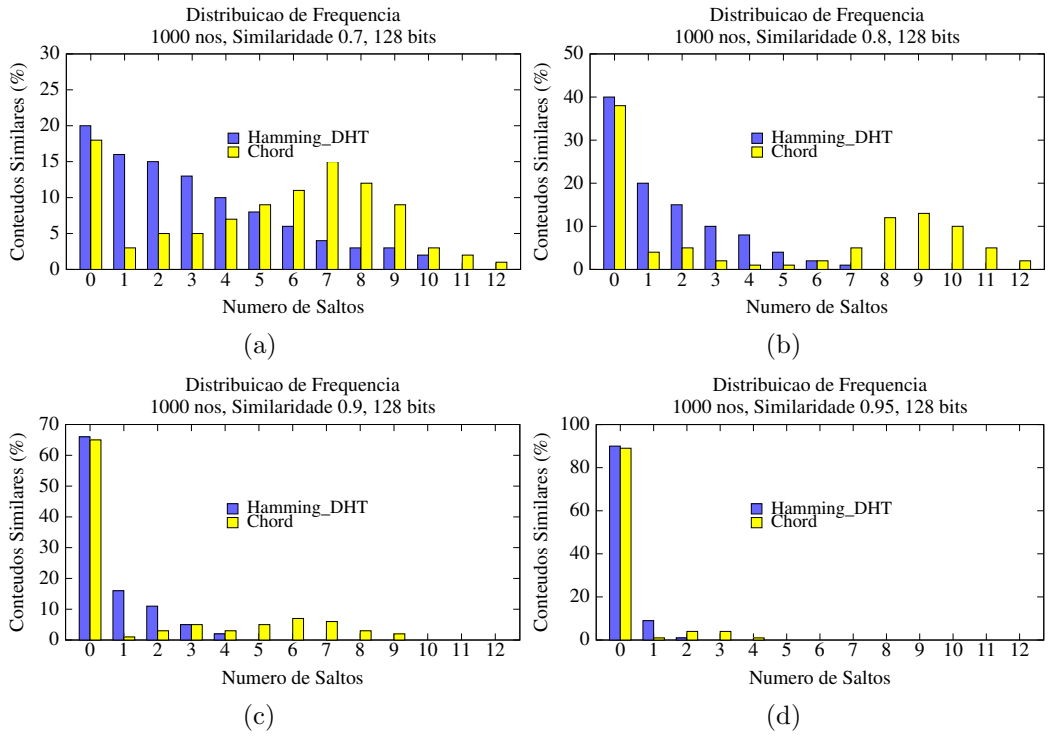


Figura 3.4: Distribuição de frequência na Hamming DHT e no Chord. 1000 nós. Níveis de similaridade de Hamming 0.7, 0.8, 0.9 e 0.95.

et al. 2003). Baseando-se na Figura 3.5(b) com similaridade de Hamming 0.8 é possível perceber que na Hamming DHT cerca de 20% e 19% dos conteúdos similares (em média) estão a 1 e 2 saltos, respectivamente, do nó de referência da consulta, p'. No Chord (Stoica et al. 2003) esses valores são 5% e 3%, respectivamente.

Uma característica que pode ser percebida nas Figuras 3.5(a) e 3.5(b) é que os resultados obtidos com o uso da Hamming DHT são mais significativos quando se usa uma profundidade de busca maior ou igual 1. Para 0 saltos, ou seja, medindo-se a quantidade de conteúdos presentes no próprio nó p' a vantagem da Hamming DHT sobre o Chord (Stoica et al. 2003) é muito pequena. Essa característica se acentua consideravelmente com o aumento do nível de similaridade requisitado.

Esse resultado não é consequência da Hamming DHT, mas sim da característica de distribuição dos identificadores similares na distância de Hamming no anel. Para explicar o ocorrido, considere uma DHT com distribuição homogênea e N nós distribuídos em um espaço de endereçamento de m *bits*, e seja $\log_2(N)$ os *bits* mais significativos desse espaço capazes de diferenciar os nós da DHT. Dado um identificador k qualquer, uma mudança de *bit* nos $(m - \log_2(N))$ bits menos significativos desse identificador mantém k', o identificador gerado com essa mudança, associado ao mesmo nó na DHT. Somente alterações nos $(m - \log_2(N))$ *bits* mais significativos provocam no identificador k' uma mudança do nó sucessor no anel. Por exemplo, supondo $N = 1024$ nós, $m = 20$ *bits*, tem-se que $\log_2(N) = 10$. Supondo ainda que esses 1024 nós estão homogeneamente espalhados no anel devido ao uso da função de *hash* MD5 para geração dos identificadores dos nós. Logo, dado um identificador k qualquer de 20 *bits*, os 1024 nós terão identificadores da seguinte forma (com alta probabilidade): $0000000000XXXXXXXXXX_2$ (1º nó

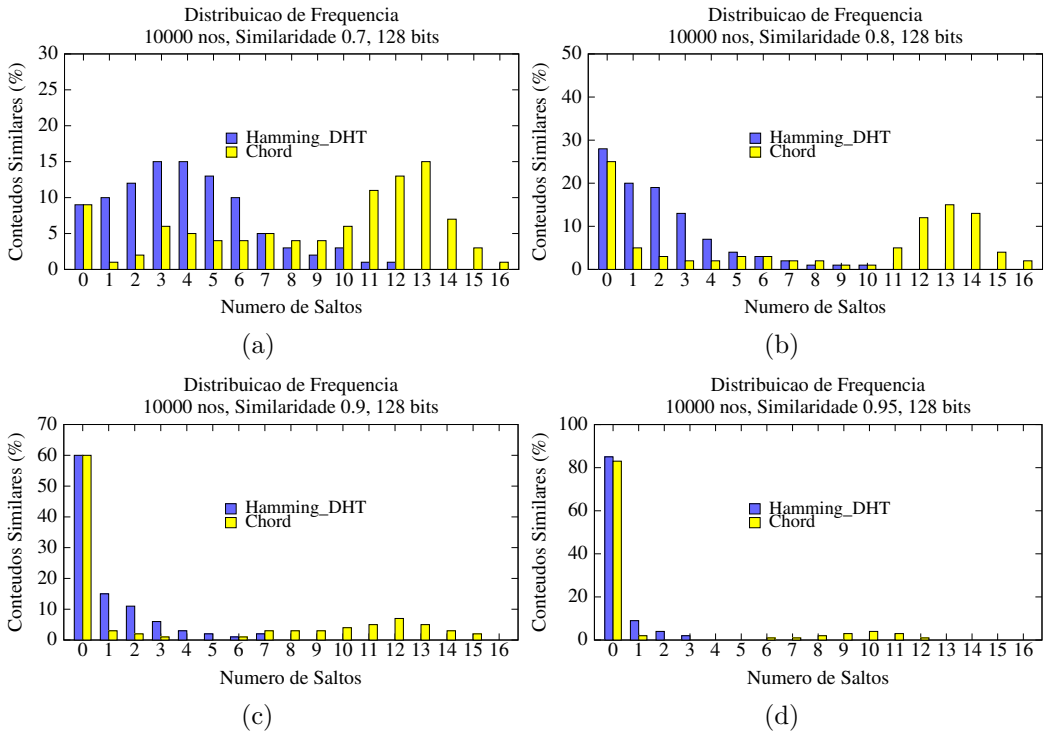


Figura 3.5: Distribuição de frequência na Hamming DHT e no Chord. 10000 nós. Níveis de similaridade de Hamming 0.7, 0.8, 0.9 e 0.95.

do anel), $0000000001XXXXXXXXXX_2$ (2º nó do anel), $0000000010XXXXXXXXXX_2$ (3º nó do anel), ... $1111111111XXXXXXXXXX_2$ (N-ésimo nó do anel), onde X é não importa (0 ou 1).

Aproximadamente a mesma quantidade de variações de *bit* ocorrem nos $m - \log_2(N)$ *bits* menos significativos do código binário natural, o que faz com que o Chord (Stoica et al. 2003) tenha uma quantidade de conteúdos similares a uma distância de 0 saltos.

Entretanto, uma característica da geração dos identificadores segundo o código de Gray binário refletido é que as alterações são sucessivamente realizadas do *bit* menos significativo para o mais significativo conforme ilustrado na Figura 3.6 (a) - fonte: (Knuth 1973). As partes escuras da figura ilustram o *bit* 1 nos identificadores segundo o código de Gray binário refletido e Gray balanceado (Figura 3.6 (b)).

Qualquer mudança nos *bits* menos significativos mantém o identificador gerado no mesmo sucessor do anel. Quando a similaridade de Hamming desejada é alta, por exemplo 0.95, somente 5% dos *bits* podem variar. Esses *bits* podem variar dentre os $(m - \log_2(N))$ *bits* menos significativos ou dentre os $\log_2(N)$ *bits* mais significativos, acarretando uma mudança de sucessor. Se o número de nós na DHT é alto, essa concentração tende a diminuir, pois $\log_2(N)$ se aproxima de $m - \log_2(N)$ aumentando a probabilidade de um conteúdo similar ser associado a outro sucessor e, consequentemente, ser armazenado em outro nó da DHT. A Figura 3.6 (b) apresenta um código binário balanceado, onde as alterações sucessivas de *bits* acontecem de forma mais equilibrada entre os *bits* mais significativos e menos significativos. Os códigos de Gray balanceados serão discutidos como trabalhos futuros ao final desse capítulo.

Em um trabalho anterior (Villaça, de Paula, Pasquini & Magalhães 2012) há resultados da

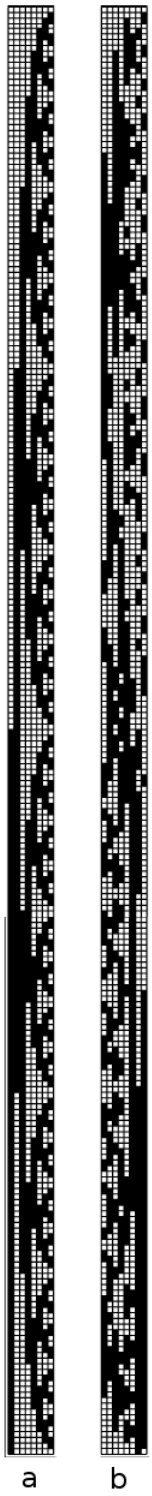


Figura 3.6: Distribuição das alterações consecutivas de bits para o código Gray binário refletido (a) e balanceado (b) - fonte: (Knuth 1973).

análise da distribuição de frequência das distâncias para identificadores de 64 bits. Estes resultados não serão apresentados nesta tese pois foram obtidos com uma base de dados sintética, gerada nas primeiras avaliações da Hamming DHT. Na tese optou-se por incluir somente os resultados com a base de dados de adultos. Entretanto, os resultados apresentam o mesmo comportamento, mostrando que o mesmo independe do tamanho do espaço de endereçamento. Nesses resultados há também avaliações com 100000 nós, limite máximo alcançado com o simulador desenvolvido. Esse limite dar-se-á em função da sobrecarga imposta pelas constantes conversões realizadas nos Algoritmos 2 e 1, o que tornava a simulação excessivamente lenta (lembrando que em situações reais essa "sobrecarga" é distribuída dentre os vários nós participantes da rede).

3.3.2 Cobertura das buscas por similaridade em função da profundidade das buscas

A Figura 3.7 apresenta os resultados da cobertura em função da profundidade de encaminhamento das consultas em uma Hamming DHT com 1000 nós. Novamente esses testes representam uma média de 10 redes distintas geradas aleatoriamente com distribuição homogênea dos nós. Os resultados são comparados com o Chord (Stoica et al. 2003). Nesses testes foram avaliados o percentual de conteúdos recuperados em relação ao total de conteúdos similares existentes na base de dados em função da profundidade das buscas. Cabe ressaltar aqui novamente que para os testes a base de dados é previamente conhecida e, portanto, sabe-se quais e quantos são os conteúdos similares para cada identificador de referência e para cada nível de similaridade desejado.

A partir dos dados da Figura 3.7(b) pode-se perceber que com uma profundidade 4 e similaridade 0.8, cerca de 91% dos conteúdos, em média, são retornados em uma busca por similaridade, enquanto que no Chord (Stoica et al. 2003) essa cobertura é de somente 51%, também em média. Novamente observa-se nos resultados de cobertura o mesmo efeito da concentração de conteúdos similares na distância 0 com o aumento do nível de similaridade nas buscas.

A Figura 3.8 apresenta os resultados da mesma avaliação de cobertura para as DHTs Hamming e Chord, ambas com 10000 nós. Considerando uma busca por similaridade com $sim_h \geq 0.9$, observa-se na Hamming DHT que, em média, 85% dos conteúdos similares foram recuperados com profundidade 2, ou seja, estavam a uma distância 2 a partir do nó de referência da consulta. Estes mesmos parâmetros no Chord (Stoica et al. 2003) produzem uma cobertura de 65%. Estes resultados podem ser observados na Figura 3.8(c).

3.4 Discussões Finais e Trabalhos Futuros relacionados à Hamming DHT

Em trabalhos de Recuperação de Informação as métricas mais usadas para verificação de desempenho são a cobertura (*recall*) e a precisão (*precision*) (Berry et al. 1999). A precisão corresponde ao percentual de conteúdos relevantes em relação ao total de conteúdos retornados em uma consulta. Neste trabalho não se avaliou a precisão visto que havia uma única base

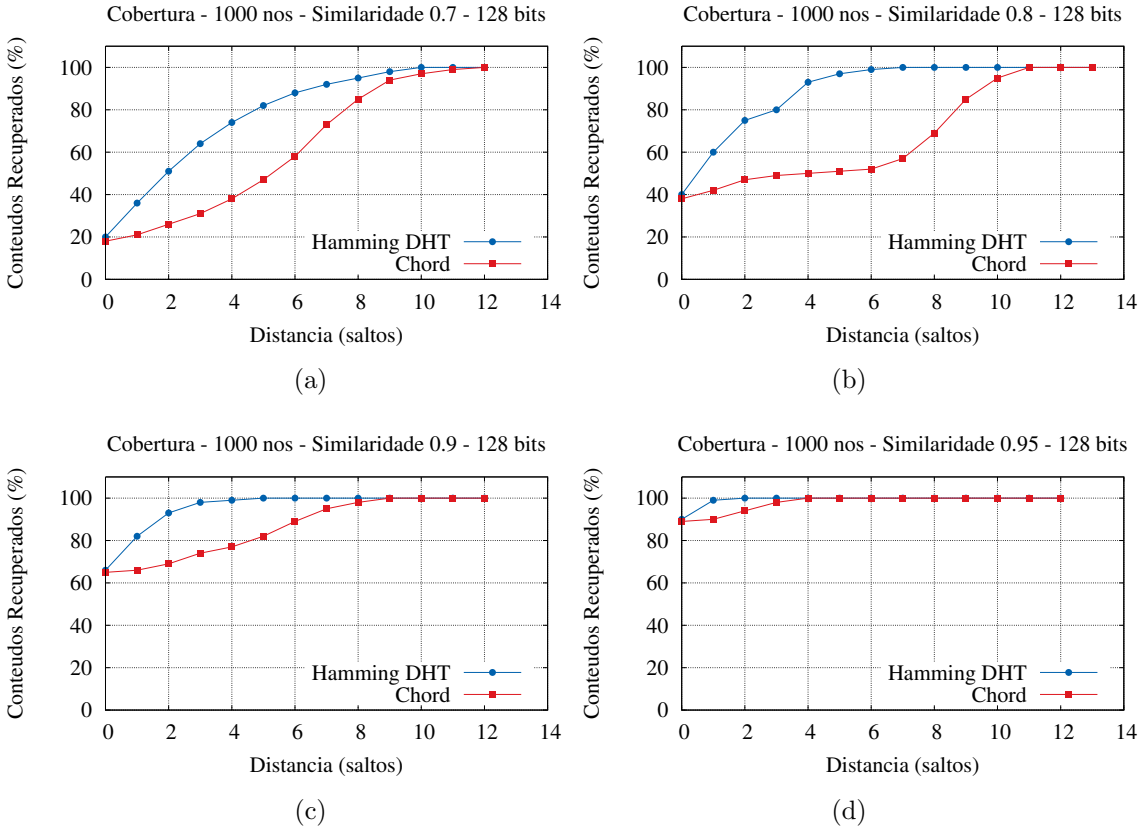


Figura 3.7: Cobertura. 1000 nós. Níveis de similaridade 0.7, 0.8, 0.9 e 0.95

de dados sendo avaliada e, portanto, todos os dados retornados na consulta são relevantes e pertencentes a essa base. Além disso, pela forma como o sistema foi construído, não é possível que objetos com similaridade menor do que aquela requerida sejam retornados, portanto, todos os conteúdos obtidos com a busca são relevantes.

Conforme observado nos resultados das avaliações realizadas com a Hamming DHT, para conteúdos similares há uma redução no número de saltos quando se compara ao Chord. Entretanto, não se pode usar a mesma prova formal do Chord para mostrar que o custo máximo do número de saltos é $O(log_2(N))$ na Hamming DHT, visto que, de acordo com a Figura 3.2, os saltos não são de comprimento logarítmico e no mesmo sentido no anel. Em casos de busca por conteúdos com baixos níveis de similaridade podem haver situações onde o número de saltos é maior do que $O(log_2(N))$.

Sabe-se que o Chord não é uma DHT especializada em cenários de busca por similaridade, o que, à primeira vista, torna injusta a comparação entre ambas as DHTs. A escolha pelo Chord deu-se pelo fato de que o mesmo pode ser tomado como referência de comparação para o pior caso (*baseline*). Além disso, por toda a inspiração que a Hamming DHT deve ao Chord (Stoica et al. 2003) na implementação de suas principais rotinas e pelo fato do Chord ser um trabalho de referência na literatura relacionada à DHTs, sendo citado em diversos trabalhos relacionados, fez-se essa escolha. Os resultados, inclusive, mostraram que essa comparação não é completamente injusta pois, considerando-se uma distância igual a 0 saltos na DHT, os resultados são bem próximos.

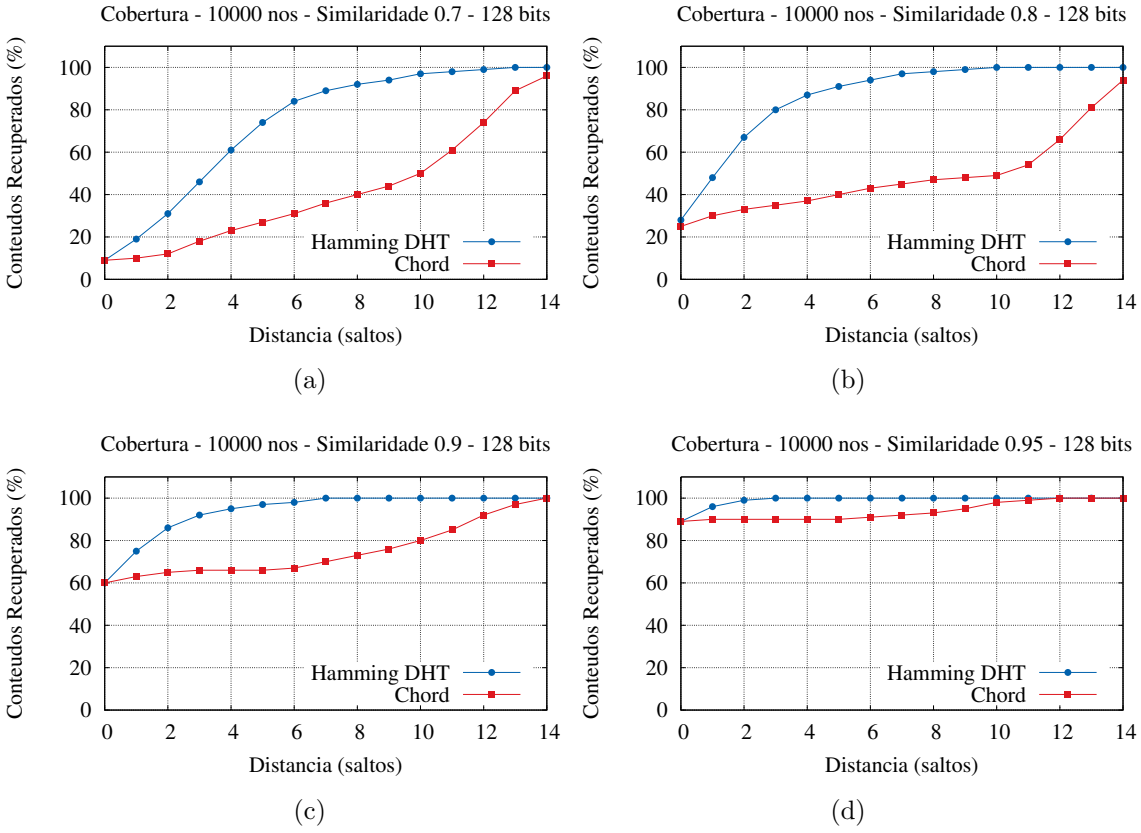


Figura 3.8: Cobertura. 10000 nós. Níveis de similaridade 0.7, 0.8, 0.9 e 0.95

Na literatura, considerando-se os trabalhos relacionados à busca por similaridade em DHT que serão apresentados no Capítulo 5, é difícil a comparação experimental desses trabalhos com a Hamming DHT. A maioria dessas propostas são experimentais, assim como a Hamming DHT e as respectivas implementações não são disponibilizadas para *download*. Além disso, em geral a metodologia e as bases de dados escolhidas são diferentes.

É de conhecimento comum que toda proposta de agregação de conteúdo em DHT leva ao desbalanceamento de carga entre os nós de uma DHT. Antes de abordar esse assunto é importante reforçar que o objetivo da Hamming DHT é reduzir as distâncias entre conteúdos similares. Com relação ao desbalanceamento deve-se considerar que a maior parte das propostas de agregação de conteúdos para a busca por similaridade utiliza a distância Euclideana, o que contribui para a concentração de conteúdos em uma determinada região do anel virtual. Pelas Figuras 3.6 e 3.3 percebe-se que identificadores similares na distância de Hamming (distância de Hamming igual a 1) espalham-se em diferentes regiões do anel virtual, e essa característica naturalmente já contribui para auxiliar o balanceamento de carga, reduzindo a concentração em determinadas regiões do anel. Por exemplo, na Hamming DHT os identificadores 1, 2, 4, 8, 16 são muito similares ao identificador 0 pois possuem distância de Hamming igual a 1. Soluções baseadas na distância Euclideana não considerariam todos esses identificadores como similares, pois o 16 é bem distante do 0. Nelas, os conteúdos similares representados por esses identificadores (1, 2, 4, 8 e 16) na Hamming DHT teriam outros valores, por exemplo: 28, 29, 31, 1 e 3. Portanto, DHTs baseadas em agregação na distância Euclideana concentrariam os

conteúdos similares ao identificador 0 na região [28..3] do anel.

A inserção de conteúdos diversos na Hamming DHT (não somente uma base de dados específica) também tende a contribuir para o equilíbrio de carga entre os nós. Nesse caso o uso de diferentes funções RHH para diferentes tipos de conteúdo também poderia ser utilizado para esse fim, por exemplo com a introdução de prefixos nos identificadores gerados por *hash* homogêneo tipo MD5 ou SHA-1 de tal forma a mapear conteúdos para nós de determinadas regiões da DHT e, internamente a essas regiões, realizar o agrupamento por similaridade de Hamming conforme estratégia definida na Hamming DHT. Essa última abordagem seria um híbrido entre DHTs homogêneas totalmente balanceadas voltadas para busca exata de conteúdos e as DHTs não-homogêneas, naturalmente desbalanceadas, porém voltadas para suportar a busca por similaridade, em uma DHT hierarquica. As regiões poderiam tornar-se ilhas temáticas conceituais na DHT aproximando-se ainda mais do trabalho desenvolvido em (de Paula, Villaça & Magalhães 2010a). Outras ações podem ser tomadas no sentido de redução do desbalanceamento, dentre elas podemos citar a realização de *caching*, replicação e estabelecimento de nós virtuais segundo a distribuição dos conteúdos no anel.

Com relação à concentração de conteúdos na distância 0, a ordenação dos identificadores segundo um código de Gray balanceado é uma característica a ser explorada em trabalhos futuros. O principal problema envolvendo os códigos de Gray balanceados é que não existem funções de conversão *grayToBinary()* e *binaryToGray()* para geração da ordenação dos identificadores no anel. Dessa forma toda a curva deveria ser gerada recursivamente e o custo computacional seria extremamente alto, inviabilizando a implementação da DHT nos dias atuais. Testes foram realizados com $m = 10$ bits, $N = 6$ e $N = 10$ nós pois em (Knuth 1973) há sequências geradas de códigos de Gray balanceados com esse tamanho. Usando-se esse código previamente gerado (pré-processado), e distribuindo-se aleatoriamente os nós da DHT nesse espaço foram feitas algumas análises da distribuição dos conteúdos. Esses resultados mostram que a concentração de conteúdos similares no sucessor do identificador de referência (distância 0) reduz em cerca de 15% nos piores casos a 50% nos melhores casos.

É importante reforçar aqui que o uso de códigos de Gray balanceados continuariam fazendo com que Chord e Hamming DHT tivessem aproximadamente a mesma quantidade de conteúdos similares à distância 0. Entretanto, essa concentração seria em menor nível e distribuindo-se melhor nas demais regiões do anel virtual, sendo também uma medida para melhorar a distribuição de carga na DHT. Testes com um número maior de bits e nós são atualmente inviáveis com o simulador desenvolvido e o código Gray balanceado.

No próximo capítulo será apresentado o HCube, uma outra proposta de arquitetura distribuída especializada na busca por similaridade. Em uma DHT, a aproximação de conteúdos similares dar-se-á no espaço lógico (virtual). Dois nós que possuem conteúdos similares e ligados por um contato em comum (*finger*) podem estar geograficamente distantes, o que implica em um aumento na latência da rede. Ou seja, reduz-se o custo dos saltos e a quantidade de acessos, mas com a contrapartida da alta latência de uma rede sobreposta. O HCube é uma arquitetura voltada para *Data Center*, cuja aproximação dar-se-á no nível físico, das ligações de rede, com a consequência imediata da redução da latência.

Capítulo 4

HCube

Neste capítulo será apresentado o HCube[1], uma proposta de infraestrutura para *Data Center* centrado nos servidores (*server-centric*) especializada no suporte a buscas por similaridade. Enquanto as soluções atuais envolvendo *Big Data* e *Data Center*, tais como o Hadoop (The Apache Software Foundation 2013) e MapReduce (Dean & Ghemawat 2008), são inquestionavelmente eficientes para lidar com aplicações tradicionais, tais como o processamento em lote de grandes volumes de dados, elas podem ser estendidas para melhor suportar a busca por similaridade.

No capítulo anterior apresentamos a Hamming DHT, uma solução de rede sobreposta voltada para suportar de maneira eficiente a busca por similaridade. Essencialmente, constatou-se que é possível atribuir identificadores para os dados representando, com um grau de precisão elevado, a similaridade entre eles. Mostrou-se, também, a possibilidade de se aproximar os dados similares em uma DHT usando as primitivas $put(k, v)$ e $get(k, sim_h)$ e um anel virtual organizado segundo a sequência do código de Gray.

Uma característica importante das redes P2P é que por serem completamente descentralizadas, elas podem ser organizadas com baixo investimento em infraestrutura. No entanto, embora os nós que armazenam dados similares possam ser alcançados com um reduzido número de saltos, não há garantia de que eles estejam fisicamente próximos na rede, uma vez que a rede é sobreposta. Essa sobreposição e a aproximação somente no nível lógico pode contribuir para o aumento da latência na recuperação de dados similares tornando-se inadequada para algumas aplicações mais exigentes.

Por esse motivo, durante o desenvolvimento dessa tese procurou-se uma alternativa capaz de aproximar os conteúdos similares na distância física na qual, de alguma forma, a similaridade de Hamming influenciasse na organização da infraestrutura da rede. Aproveitando-se do fato de que os *Data Centers* (DC) são excelentes alternativas no processamento de grandes volumes de dados (Agrawal, El Abbadi, Antony & Das 2010), o objetivo do HCube (*Hamming Cube*) é viabilizar o armazenamento de conteúdos similares em servidores fisicamente próximos, acelerando a busca por similaridade.

[1]Durante o processo de revisão de um dos artigos do HCube fomos alertados, corretamente, de que a denominação de cubo usada no HCube não é adequada, visto que suas dimensões não precisam ter, necessariamente, o mesmo tamanho. Formalmente o HCube representa a projeção de um hipercubo em um espaço tridimensional com redução de dimensões. Entretanto, optou-se por manter o nome HCube na tese por compatibilidade com as publicações já realizadas.

A seguir, este capítulo detalha a proposta do HCube através da descrição da sua arquitetura e operação, assim como, os resultados obtidos em *Data Centers* com 64, 128, 256, 512, 1024, 2048, 4096 e 8192 servidores.

4.1 Apresentação do HCube

As ferramentas básicas para a construção do HCube são: a função de *hash* RHH; a representação vetorial dos conteúdos armazenados no *Data Center*; o uso da similaridade do cosseno e da similaridade de Hamming entre os identificadores dos vetores de conteúdos e dos servidores do HCube; o modelo BCube (Guo, Lu, Li, Wu, Zhang, Shi, Tian, Zhang & Lu 2009) na qual os servidores são usados tanto para armazenamento de dados quanto para roteamento e encaminhamento das consultas; o roteamento XOR (Pasquini 2011) e (Pasquini, Verdi & Magalhães 2011); e a organização dos servidores de acordo com a curva de Gray em um hipercubo tridimensional.

Inspirando-se na proposta da Hamming DHT, no HCube os servidores são identificados e organizados de acordo com a sequência do código de Gray, onde cada servidor está conectado a outros servidores cujos identificadores apresentam distância de Hamming igual a 1. Mais especificamente, em se tratando de um espaço tridimensional, a sequenciação dos servidores segue a curva SFC de Gray, aproximando na distância física os servidores com identificadores vizinhos na distância de Hamming.

Entretanto, diferentemente da Hamming DHT, no HCube os servidores possuem um espaço de endereçamento diferente do espaço de endereçamento dos conteúdos e, portanto, não é mais possível atribuir a responsabilidade pelo armazenamento dos identificadores ao nó que o sucede no espaço de endereçamento. Dessa forma, necessita-se de um mapeamento consistente entre o identificador do conteúdo e o endereço do respectivo servidor que será responsável pelo seu armazenamento. Em geral, o espaço de identificação dos conteúdos possui um comprimento muito maior do que o espaço de endereçamento dos servidores, visto que existe uma quantidade enorme de conteúdos nos cenários de *Big Data* e muitas limitações de infraestrutura relacionadas ao número de servidores em um *Data Center*. Conforme já visto nesta tese, as funções de *hash* possuem a capacidade de reduzir o espaço de endereçamento de maneira eficaz. Mais ainda, sabe-se que a função RHH corresponde a uma família de funções LSH capaz de representar a similaridade do cosseno entre vetores de conteúdo na distância de Hamming dos respectivos identificadores gerados por essa função.

A hipótese levantada durante a proposição do HCube foi: será que os identificadores de conteúdos com comprimento $m = 128$ *bits* podem ser reduzidos para um número menor de *bits*, por exemplo $m = 10$ *bits*, usando a função RHH e, ainda assim, preservar a similaridade do cosseno entre os vetores? Nessa redução é necessário manter a similaridade do cosseno entre os identificadores originais, representados por vetores binários, onde cada *bit* é uma dimensão do vetor, na similaridade de Hamming entre os identificadores reduzidos. Em outras palavras, considerando que identificadores binários de $m = 128$ *bits* (c_1 e c_2) podem ser representados por vetores binários de 128 dimensões, quanto menor a distância de Hamming entre c_1 e c_2, maior será a probabilidade de que c_1 e c_2 sejam reduzidos para o mesmo identificador com $m = 10$ *bits*, ou para identificadores reduzidos que possuam baixas distâncias de Hamming entre si.

Vale reforçar que se c_1 e c_2 são representados como vetores binários e usados como entrada para a função RHH durante o processo de redução de dimensões, usa-se a similaridade do cosseno entre c_1 e c_2 para determinação da probabilidade de mapeamento no identificador reduzido.

Esse identificador reduzido corresponde, na proposta do HCube, ao identificador do servidor responsável pelo armazenamento do conteúdo representado pelo identificador original com $m = 128$ *bits*.

A principal contribuição do HCube é mostar como a curva SFC de Gray, utilizada no sequenciamento dos identificadores dos servidores no cubo tridimensional, pode contribuir para a redução da distância física entre conteúdos similares e, desta forma, contribuir para auxiliar a busca por similaridade.

4.1.1 Camadas do HCube

Esta seção fornece uma descrição geral do HCube baseada em uma organização lógica dividida em duas camadas com funções distintas: a camada de admissão e a camada de armazenamento.

A camada de admissão fornece a interface entre o mundo exterior e o HCube. Essa camada é composta por um conjunto de servidores de admissão que operam no topo do HCube recebendo as consultas dos usuários/aplicações e preparando-as para serem injetadas no HCube. Para simplificar, a figura mostra apenas um servidor de admissão. Entretanto, a escolha desse servidor pode ser gerenciada em função do balanceamento de carga de processamento entre os servidores do HCube. As camadas são apresentadas na Figura 4.1.

No canto superior esquerdo da Figura 4.1 é apresentado um vetor consulta $\overrightarrow{q}$ composto por n dimensões sendo admitido pelo HCube associado a um nível de similaridade de Hamming sim_h. Uma vez que o vetor de consulta $\overrightarrow{q}$ é admitido, o servidor de admissão utiliza a função RHH para gerar o identificador dos dados di usado como referência para a busca por similaridade. Na sequência, o servidor de admissão reduz o identificador di, também usando a função RHH, a fim de se obter o identificador do servidor de hospedagem si responsável pelo armazenamento de di. Neste exemplo, o si tem um comprimento de 6 *bits*, considerando um HCube composto por 64 servidores (2^6 identificadores de servidores no espaço) e o di tem um comprimento de 128 *bits*.

Após a admissão do vetor $\overrightarrow{q}$, o identificador dos dados de referência di e o nível de similaridade desejado sim_h são encaminhados, usando roteamento XOR (Seção 4.1.3), a partir do servidor de admissão para o servidor de hospedagem si. O servidor de hospedagem deve procurar em sua base de dados local (camada de armazenamento) todos os identificadores de conteúdo que obedeçam ao nível de similaridade de Hamming desejado para a consulta e, assim como na Hamming DHT, a partir do servidor de hospedagem si uma série de mensagens $get(di,sim_h)$ são encaminhadas para os servidores vizinhos no DC com o objetivo de aumentar a cobertura da busca. Especificamente para fins de avaliação, as mensagens $get(di,sim_h)$ são gradualmente enviadas a todos os servidores vizinhos a si, e para os vizinhos dos vizinhos, segundo uma ordem crescente da distância de Hamming entre os identificadores dos servidores. Ou seja, a partir de si encaminha-se a busca por similaridade para os seus vizinhos físicos no HCube, si', que possuem distância de Hamming igual a 1 (um) com si. Cada si' repassa a

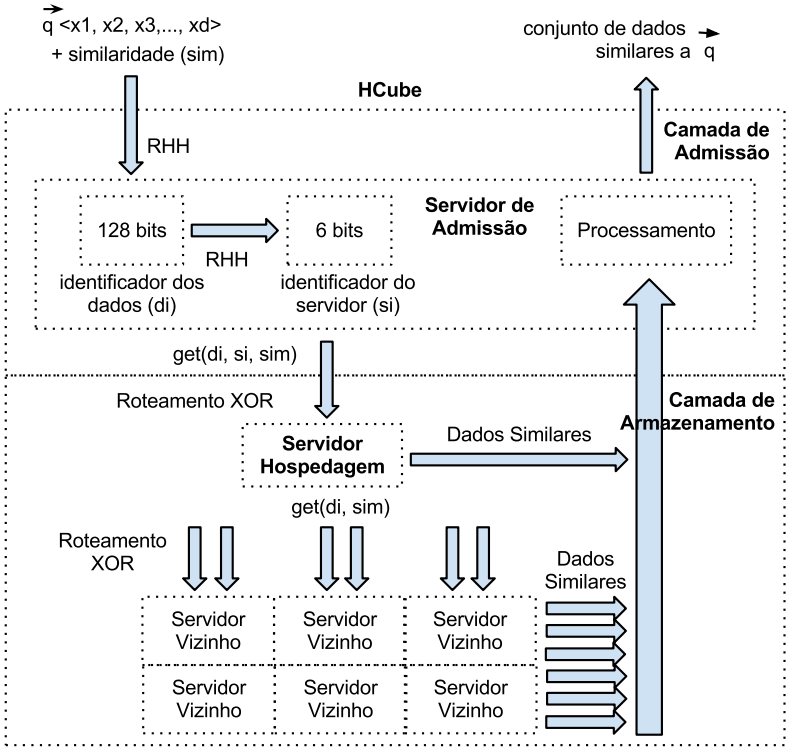


Figura 4.1: Operação do HCube em camadas. Exemplo durante uma busca por similaridade.

consulta a seus vizinhos físicos si'' que possuem distância de Hamming igual a 1 (um) com si' e, consequentemente, igual a 2 (dois) com si, e assim sucessivamente. O objetivo dessa avaliação é fornecer uma análise completa da distância existente entre os servidores que armazenam dados similares e a cobertura oferecida pelas buscas.

Conforme as mensagens *get* vão sendo encaminhadas para outros servidores dentro do HCube na camada de armazenamento, os servidores que contêm dados similares devem retornar estes dados para o servidor de admissão, responsável pelo tratamento das respostas. O servidor de admissão agrupa as respostas e entrega o conjunto de dados similares para o usuário/aplicação, concluindo o processo. Como uma implementação alternativa, o servidor de admissão pode devolver uma lista de referências para os dados similares ao invés de devolver todo o conjunto de dados, evitando a movimentação desnecessária de grandes massas de dados de maneira imediata. Essa implementação permite a recuperação gradual dos dados similares, por exemplo, filtrando-as por data de criação e/ou armazenamento.

4.1.2 Organização dos servidores no HCube

O HCube é composto por um conjunto de servidores organizados em uma topologia de hipercubo tridimensional, onde tais servidores têm papel fundamental no DC. Nesse contexto, equipamentos de rede de baixo custo, COTS (*Commodity Off-the-Shelf*), podem ser utilizados para simplificar o processo de cabeamento do DC. Até mesmo ligações diretas podem ser estabelecidas entre os servidores. A característica mais importante do sistema centrado no servidor (*server-centric*) é que os elementos de rede (por exemplo, *switches*) não estão envolvidos nas de-

cisões de encaminhamento de tráfego, eles simplesmente fornecem conexões entre os servidores, que são responsáveis por todas as decisões de encaminhamento.

Cada servidor no HCube dispõe de um processador de propósito geral, memória, armazenamento persistente e seis placas de rede (nomeadas de `eth0` a `eth5`) distribuídas em três eixos (x, y e z). Cada servidor usa as placas de rede `eth0` e `eth1` no eixo x, `eth2` e `eth3` no eixo y, `eth4` e `eth5` no eixo z. A fim de proporcionar um maior número de rotas alternativas dentro do HCube, aumentar a tolerância a falhas e reduzir a distância entre pares de servidores, a topologia proposta conecta as extremidades do HCube em todos os três eixos. A Figura 4.2 apresenta uma topologia do HCube composta por 8 servidores. Observe nesta figura a existência de seis interfaces de rede em todos os servidores, as ligações nos três eixos e as conexões entre os servidores nas extremidades.

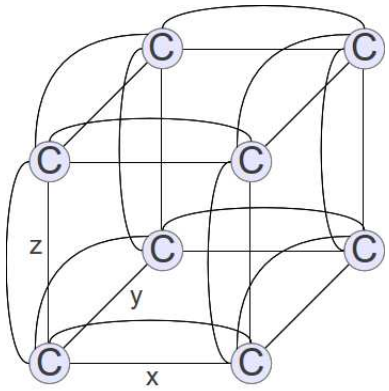


Figura 4.2: Ligações x,y,z de um HCube com 8 servidores.

Conforme já fora dito anteriormente, a fim de facilitar a busca por similaridade o HCube adota o código de Gray para organizar e alocar os identificadores dos servidores. Dessa forma, os identificadores de servidores vizinhos diferem em apenas um *bit* e, consequentemente, tendem a armazenar conteúdos similares indexados usando a função RHH. A principal contribuição desta tese relativamente à arquitetura do HCube consiste na utilização da curva de preenchimento de Gray como forma de agregar dados similares em servidores vizinhos na distância de Hamming. A Figura 4.3 apresenta um HCube com 64 servidores (identificadores de 6 bits). A visualização do HCube está organizada em quatro níveis (L1, L2, L3 e L4) para efeito de simplicação. Esses níveis representam planos no eixo z.

Na Figura 4.3 considere o servidor 29 (011101_2) localizado no nível L2. Os seis vizinhos deste servidor são: 28 (011100_2, L2) e 31 (011111_2, L2) no eixo x, 25 (011001_2, L2) e 21 (010101_2, L2) no eixo y; 13 (001101_2, L1) e 61 (111101_2, L3) no eixo z. Note que todos estes vizinhos apresentam identificadores cuja distância de Hamming para o servidor 29 é igual a 1.

Para exemplificar as ligações entre os servidores nas extremidades, considere o servidor 60 (111100_2), localizado no nível L3. Os seis vizinhos deste servidor são: 62 (111110_2, L3) e 61 (111101_2, L3) no eixo x, 52 (110100_2, L3) e 56 (111000_2, L3) no y eixo; 44 (101100_2, L4) e 28 (011100_2, L2) no eixo z. Todos eles também apresentam identificadores com distância de Hamming igual a 1 com o servidor 60.

O HCube formado por 64 servidores, conforme as matrizes apresentadas, é considerado

$$L1 = \begin{bmatrix} 8 & 9 & 11 & 10 \\ 12 & 13 & 15 & 14 \\ 4 & 5 & 7 & 6 \\ 0 & 1 & 3 & 2 \end{bmatrix} \qquad L3 = \begin{bmatrix} 56 & 57 & 59 & 58 \\ 60 & 61 & 63 & 62 \\ 52 & 53 & 55 & 54 \\ 48 & 49 & 51 & 50 \end{bmatrix}$$

$$L2 = \begin{bmatrix} 24 & 25 & 27 & 26 \\ 28 & 29 & 31 & 30 \\ 20 & 21 & 23 & 22 \\ 16 & 17 & 19 & 18 \end{bmatrix} \qquad L4 = \begin{bmatrix} 40 & 41 & 43 & 44 \\ 44 & 45 & 47 & 46 \\ 36 & 37 & 39 & 38 \\ 32 & 33 & 35 & 34 \end{bmatrix}$$

Figura 4.3: Visualização dos níveis no eixo z de um HCube com 64 servidores.

como um caso especial uma vez que o número de interfaces de rede dos servidores (6 interfaces) corresponde exatamente ao comprimento dos identificadores (6 *bits*). Neste caso particular, todos os servidores cujos identificadores apresentam distância de Hamming igual a 1 são vizinhos físicos no HCube.

Por outro lado, em maiores espaços de identificação de servidores, alguns dos servidores com distância de Hamming igual a 1 não estarão diretamente conectados. Basicamente, seis servidores com distância de Hamming igual a 1 estarão diretamente ligados e os demais servidores serão posicionados em algum ponto mais distante do HCube aumentando a distância física (maior número de saltos) entre esses servidores.

A Figura 4.4 mostra os níveis de um HCube com 128 servidores, 8x4x4 (7 *bits*). Nesta figura, a sequência da Gray SFC é representada pelas setas. Considere o servidor 9 (0001001_2) localizado no nível L1. Com identificadores de 7 *bits*, há 7 servidores com distância de Hamming igual a 1 para com o identificador do servidor 9. Eles estão organizados da seguinte forma: os servidores 8 (0001000_2, L1) e 11 (0001011_2, L1) no eixo x; servidores 25 (0011001_2, L1) e 1 (0000001_2, L1) no eixo y; servidores 41 (0101001_2, L2) e 73 (1001001_2, L4) no eixo z e, finalmente, o servidor 13 (0001101_2, L1) que não é vizinho físico do servidor 9 e está localizado em uma distância maior no HCube. É importante destacar que a distância em saltos para o servidor 13 é encurtada em função das ligações estabelecidas entre os servidores das extremidades do HCube mas o seu valor numérico é dado em função das tabelas de roteamento dos servidores, geradas pelo mecanismo do XOR, apresentado na próxima seção desse capítulo.

Um HCube completo deveria ter m dimensões, onde m é o número de *bits* nos identificadores dos servidores, de tal forma a oferecer os melhores resultados em uma busca utilizando a similaridade de Hamming. No entanto, um HCube com muitas dimensões é muito difícil de ser construído, distante da realidade das estruturas de cabeamento e ligações entre servidores em um DC. Desta forma, a estrutura tridimensional do HCube oferece um bom compromisso entre a complexidade do cubo e os resultados obtidos em uma busca por similaridade.

4.1.3 Roteamento XOR

Quando o cubo é completo, por exemplo com $m = 6$ *bits*, todos os 64 identificadores de servidores com distância de Hamming 1 são vizinhos físicos. Desta forma o roteamento para encaminhar as mensagens é natural, como acontece em qualquer hipercubo. Entretanto, conforme

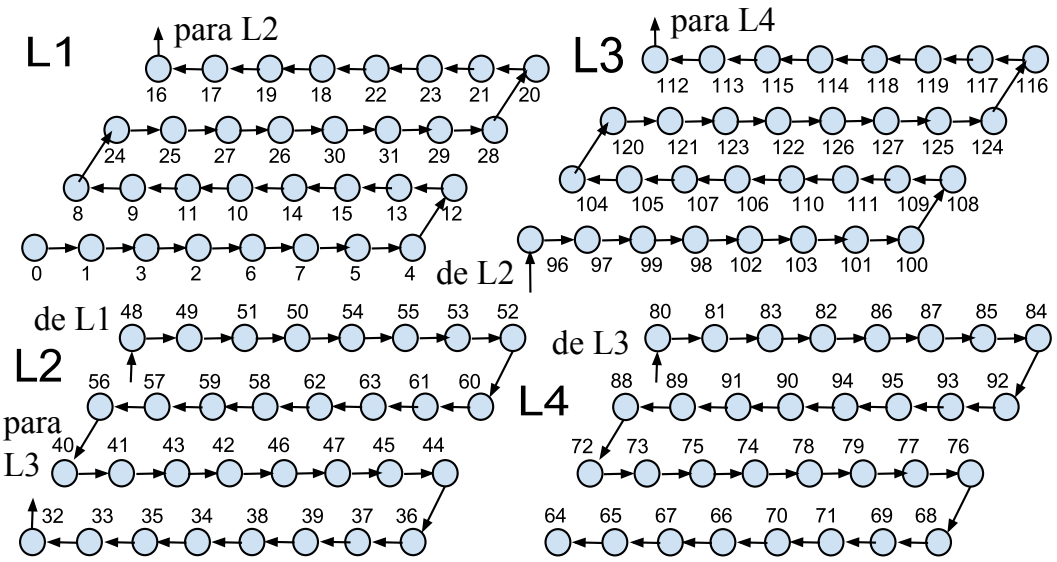


Figura 4.4: Curva de Gray em um HCube de dimensões 8x4x4 (128 servidores).

já discutido anteriormente nesta seção, à medida em que se aumenta o espaço de identificação dos servidores nem todos os identificadores vizinhos com distância de Hamming 1 estarão diretamente conectados. Desta forma, para se encaminhar as mensagens para alcançar estes servidores vizinhos na distância de Hamming, porém distantes no HCube, torna-se necessário o uso de alguma estratégia de roteamento.

O roteamento baseado em ou-exclusivo, XOR, foi proposto em (Pasquini 2011) e, conforme mostrado em (Villaça, Pasquini, de Paula & Magalhães 2013b), é bem adaptado ao cenário proposto no HCube. O roteamento XOR usa identificadores planos de m bits para organizar suas tabelas de roteamento em m colunas e rotear mensagens na rede. Seu algoritmo de roteamento usa a operação de OU-exclusivo *bit* a *bit* (XOR) entre dois identificadores k_1 e k_2 para representar a distância entre k_1 e k_2, denotada por:

$$d(k_1, k_2) = k_1 \oplus k_2$$

sendo $d(k, k) = 0$ e $d(k_1, k_2) > 0$, $\forall k_1 \neq k_2$.

Dada uma mensagem originada no servidor k_1 e destinado ao servidor k_2, e denotando $\mathbb{Y}$ como o conjunto dos identificadores presentes na tabela de roteamento de k_1, o mecanismo de roteamento XOR aplicado no servidor k_1 seleciona um servidor $k \in \mathbb{Y}$ que minimiza a distância k_1 até k_2.

A fim de suportar o encaminhamento de tráfego no interior do HCube, as tabelas de roteamento baseadas em XOR mantidas em cada servidor são formadas por $O(m)$ linhas. Em cada linha dessa tabela, o conhecimento sobre os servidores vizinhos está dividido em m colunas, chamadas de *buckets*, e representadas por $\beta_i, 0 \leq i \leq m-1$. Cada vez que um servidor k_1 obtém informações sobre um servidor k ele armazena as informações sobre k na coluna β_{m-1-i} tal que i é o maior valor que satisfaz a seguinte condição:

$d(k_1, k)$ div $2^i = 1$, $k_1 \neq k$, $0 \leq i \leq m-1$

Para exemplificar a criação de tabelas de roteamento, considere o exemplo dado na Seção 4.1.2 com um HCube de 64 servidores e $m = 6$ bits. Assumindo o servidor 29 (011101_2) como k_1 e seu vizinho 13 (001101_2) como k, a distância $d(k_1, k) = 010000_2$ e o maior i que satisfaz a condição é $i = 4$. Concluindo, o identificador $k = 13$ (001101_2) deve ser armazenado na coluna $\beta_{m-1-i} = \beta_{6-1-4} = \beta_1$. Basicamente, a condição indica que o servidor k_1 armazena o conhecimento sobre o servidor k na coluna β_{m-1-i}, onde $m-1-i$ é o comprimento do prefixo mais longo em comum existente entre os dois identificadores. Isto pode ser observado na Tabela 4.1, onde $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ armazenam os identificadores com *lcp* (*longest common prefix*, ou prefixo mais longo em comum) de comprimentos 0, 1, 2, 3, 4 e 5, relativos ao servidor 29 (011101_2).

Tabela 4.1: Tabela de roteamento hipotética para o servidor 29 (011101_2) em um HCube usando um espaço de identificação de servidores onde $m = 6$.

β_0	β_1	β_2	β_3	β_4	β_5
111101	**0**01101	**01**0101	**011**001	**0111**11	**01110**0
100000	**0**00000	**01**0000	**011**000	**0111**10	
100010	**0**00100	**01**0100	**011**010		
111111	**0**01111	**01**0111	**011**011		

Além disso, embora a informação não esteja visualmente incluída na Tabela 4.1, na implementação do algoritmo de roteamento XOR torna-se necessário acrescentar a informação sobre a interface de rede de saída para cada entrada na tabela de roteamento dos servidores do HCube. Tomando-se o mesmo servidor 29 (011101_2) como exemplo, e considerando-se o *bucket* β_4, nota-se que há uma referência para o servidor 30 (011110_2). Essa entrada para o servidor 30 (011110_2) deverá ser acrescentada da informação "`eth0`", que corresponde à interface da esquerda do servidor 29 (011101_2) na direção do eixo x (vide L1 na Figura 4.3). Em outras palavras, para se alcançar o servidor 30 (011110_2) a partir do servidor 29 (011101_2) deve-se utilizar a interface `eth0`.

Tal abordagem de roteamento é uma das principais vantagens do mecanismo baseado no XOR, uma vez que cada servidor precisa conhecer apenas um vizinho por *bucket*, dentre os possíveis 2^m servidores disponíveis no HCube, para ser capaz de encaminhar todas as mensagens com sucesso. Se um servidor tiver mais de uma entrada por *bucket*, tais entradas adicionais podem otimizar o processo de roteamento ajudando-o a encontrar melhores rotas.

Além disso, assumindo a distribuição dos identificadores segundo a curva de Gray adotada no HCube, o preenchimento das colunas da tabela de roteamento dos servidores é facilitado, pois cada vizinho com distância de Hamming igual a 1 encaixa-se exatamente em uma coluna da tabela. Como exemplo, veja na primeira linha da Tabela 4.1 a presença de todos os servidores cujos identificadores apresentam distância de Hamming igual a 1 com o servidor 29 (011101_2). Essa característica reduz a troca de mensagens de descoberta de vizinhos no HCube durante a inicialização de cada servidor. Como mencionado anteriormente, para um HCube com mais

de 64 servidores, apenas seis vizinhos com distância Hamming iguais a 1 estarão conectados fisicamente. Neste caso, um processo de sinalização deve ser usado para descobrir os demais servidores e preencher o restante da tabela. Esse processo está descrito em (Pasquini 2011).

4.2 Avaliação do HCube

Para avaliar o HCube foi utilizado o ambiente de simulação de rotamento XOR desenvolvido em (Pasquini 2011). Para geração dos identificadores de servidor de hospedagem si e conteúdo di utilizou-se a função de *hash* RHH modificada a partir da implementação realizada em (de Paula et al. 2010b) para adequação às necessidades desta tese. Para atribuição dos identificadores aos servidores e organização do HCube desenvolveu-se um gerador da curva de Gray usando a linguagem de programação Java. Por fim, a interface com o usuário possui duas primitivas básicas baseadas no mesmo modelo chave-valor descrito na Hamming DHT: $put(k, v)$ e $get(k, sim_h)$. A primeira permite que um valor (um dado, ou um conteúdo, representado por v) seja armazenado no HCube associando-o a uma chave (identificador) k. A segunda parte permite que um identificador de referência, k, seja fornecido ao HCube junto a um nível de similaridade de Hamming desejado, sim_h.

Para avaliar o HCube os seguintes experimentos foram realizados:

- Usando a função de *hash* RHH foram gerados os identificadores de 128 *bits* para todos os 48842 perfis de adultos na base de dados;
- Esses identificadores foram reduzidos para 6, 10, 11, 12 e 13 bits e distribuídos em HCube com 64, 1024, 2048, 4096 e 8192 servidores, organizados de acordo com a sequência da curva de Gray, a sequência binária natural e uma distribuição aleatória, sendo essas duas últimas somente para efeito de comparações entre os resultados;
- Os mesmos 48842 perfis de adultos presentes na base de dados foram usados como vetores de consulta nos HCube considerando-se níveis de similaridade de Hamming maiores ou iguais a 0.7, 0.8, 0.9 e 0.95;
- A partir do servidor de hospedagem todos os perfis que se encaixavam dentro do nível de similaridade de Hamming fornecido foram recuperados, avaliando-se a distância e a cobertura da busca nas três organizações sugeridas.

Esses testes nos permitiram destacar os seguintes aspectos:

- A correlação entre a similaridade do cosseno e a similaridade de Hamming dos identificadores dos servidores de hospedagem (6, 10, 11, 12 e 13 bits) gerados a partir do identificador do conteúdo, de 128 bits;
- A distância média entre dados similares no HCube, contada pelo número de saltos entre os seus respectivos servidores de hospedagem, em função dos níveis de similaridade e dos três HCube testados (organização segundo a curva de Gray, sequencial e aleatória);

- A distribuição de frequência do número de saltos necessários para recuperar todos os perfis similares de acordo com a distância de Hamming dos identificadores nas três organizações de HCube;

- A cobertura em função da profundidade das buscas para cada nível de similaridade nos três HCube. Nesse teste a busca era expandida para os vizinhos do servidor de hospedagem (si) e para os vizinhos dos vizinhos, de forma a relacionar a profundidade da busca e a quantidade de conteúdos recuperados.

Para realizar os testes, perfis de adultos foram indexados e distribuídos no HCube usando a primitiva $put(k,v)$. A primitiva $get(k,sim_h)$ foi usada para recuperar os perfis similares, onde k é o identificador de um perfil de adulto de referência e sim_h é o nível de similaridade de Hamming desejado. Conforme já dito anteriormente, apenas com o objetivo de avaliar a relação entre a cobertura total da consulta e o número de saltos necessários, a busca foi estendida a todos os vizinhos do servidor de hospedagem com distâncias físicas maiores do que 1. Em regime de operação no DC este valor de profundidade da busca pode ser considerado um parâmetro configurável em função da carga nos servidores e da cobertura desejada para as buscas.

Além disso, devido à enorme quantidade de pares de vetores para serem analisados (48842^2 = 2385540964), limitou-se a avaliação dos resultados em 10% dos pares em cada nível de similaridade (0.7, 0.8, 0.9 e 0.95). Apesar desta redução aplicou-se o teste de intervalo de confiança estatística de 95% nos resultados e verificou-se que os mesmos foram satisfatórios.

4.2.1 Correlações

Conforme descrito na Figura 4.1, os identificadores dos servidores de hospedagem, si, são gerados a partir do identificador do conteúdo de 128 *bits*, di, tomando-se o mesmo como um vetor de 128 dimensões com atributos binários. O objetivo dos testes é mostrar que a similaridade do cosseno, mantida na similaridade de Hamming dos identificadores de conteúdo de 128 *bits* (Seção 2.3), é preservada após a redução do identificador di e a geração dos identificadores si. A Tabela 4.2 mostra a correlação entre a similaridade do cosseno e similaridade de Hamming dos identificadores dos servidores de hospedagem de 6, 10, 11, 12 e 13 *bits*, gerados a partir do identificador de 128 *bits*. Os resultados são classificados segundo os níveis de similaridade 0.7, 0.8, 0.9 e 0.95.

Tabela 4.2: Correlação média entre as similaridades do cosseno e de Hamming.

sim_{cos}	6-bits sim_{ham}	10-bits sim_{ham}	11-bits sim_{ham}	12-bits sim_{ham}	13-bits sim_{ham}
≥ 0.7	0.37	0.46	0.49	0.51	0.58
≥ 0.8	0.42	0.57	0.61	0.68	0.78
≥ 0.9	0.69	0.74	0.77	0.84	0.90
≥ 0.95	0.96	0.97	0.98	0.98	0.99

Para os níveis de similaridade mais altos (≥ 0.9), os resultados mostram uma correlação forte (> 0.8) entre a similaridade do cosseno e a similaridade de Hamming dos identificadores dos seus respectivos servidores de hospedagem no HCube. Quando o nível de similaridade cai abaixo de 0.9, é possível notar uma correlação moderada (entre 0.4 e 0.8) entre as similaridades dos servidores, exceto para o nível de similaridade 0.7 e HCube com 64 servidores (identificadores de 6 *bits*). Conclui-se desta tabela que quanto maior é o nível de similaridade e o tamanho do identificador *si*, maior é a correlação entre a similaridade do cosseno e a similaridade de Hamming dos servidores de hospedagem dos perfis de usuários no HCube.

4.2.2 Distribuição das distâncias entre servidores de hospedagem

As avaliações realizadas nesta seção mostram a relação entre as distâncias de Hamming dos identificadores dos servidores de hospedagem e a distância (medida em número de saltos) entre eles nas três configurações: Gray, Sequencial e Aleatório. Os resultados da avaliação estão presentes na Figura 4.5.

A partir da Figura 4.5 é possível perceber que a distância média entre os servidores no HCube, medida em número de saltos, é diretamente proporcional à distância de Hamming dos seus identificadores. Na organização sequencial percebe-se que o número de saltos para distâncias de Hamming baixas (o que implica em maior similaridade) é sempre maior do que na organização segundo a curva de Gray, pois esta curva é especializada no agrupamento segundo a similaridade de Hamming. Entretanto, esse comportamento se inverte ao atingirmos a metade, ou mais, do tamanho dos identificadores utilizados. Por exemplo, com 10 *bits* de identificador de servidor (Figura 4.5(b)) e distância de Hamming igual a 7 ($sim_{ham} = 0.3$), observa-se que a organização de Gray oferece aproximadamente 11.5 saltos de distância, enquanto que a organização sequencial oferece aproximadamente 9.5, compensando o efeito de agrupamento por distância de Hamming da curva de Gray. Esse comportamento justifica a vocação do HCube para buscas por similaridade de Hamming. Na organização aleatória percebe-se um equilibrio entre todas as faixas de distâncias de Hamming uma vez que nenhum agrupamento é fornecido.

4.2.3 Distância média entre conteúdos similares

As médias do número de saltos necessários para recuperar os perfis de adultos no DC em função do nível de similaridade desejado estão apresentadas nas Tabelas 4.3 (HCube), 4.4 (organização sequencial) e 4.5 (organização aleatória). Foram avaliados DCs com 64 ($m = 6$ *bits*), 1024 ($m = 10$ *bits*), 2048 ($m = 11$ *bits*), 4096 ($m = 12$ *bits*) e 8192 ($m = 13$ *bits*) servidores usando a proposta do HCube (cubo tridimensional e sequência do código de Gray) e usando o mesmo cubo com identificadores organizados de maneira sequencial natural e aleatória. A implementação sequencial segue o mesmo formato da curva da Figura 4.4, entretanto os identificadores são organizados em sua sequência natural $[0, 1, 2...2^m - 1]$ onde m é o número de *bits* do identificador dos servidores que formam o HCube.

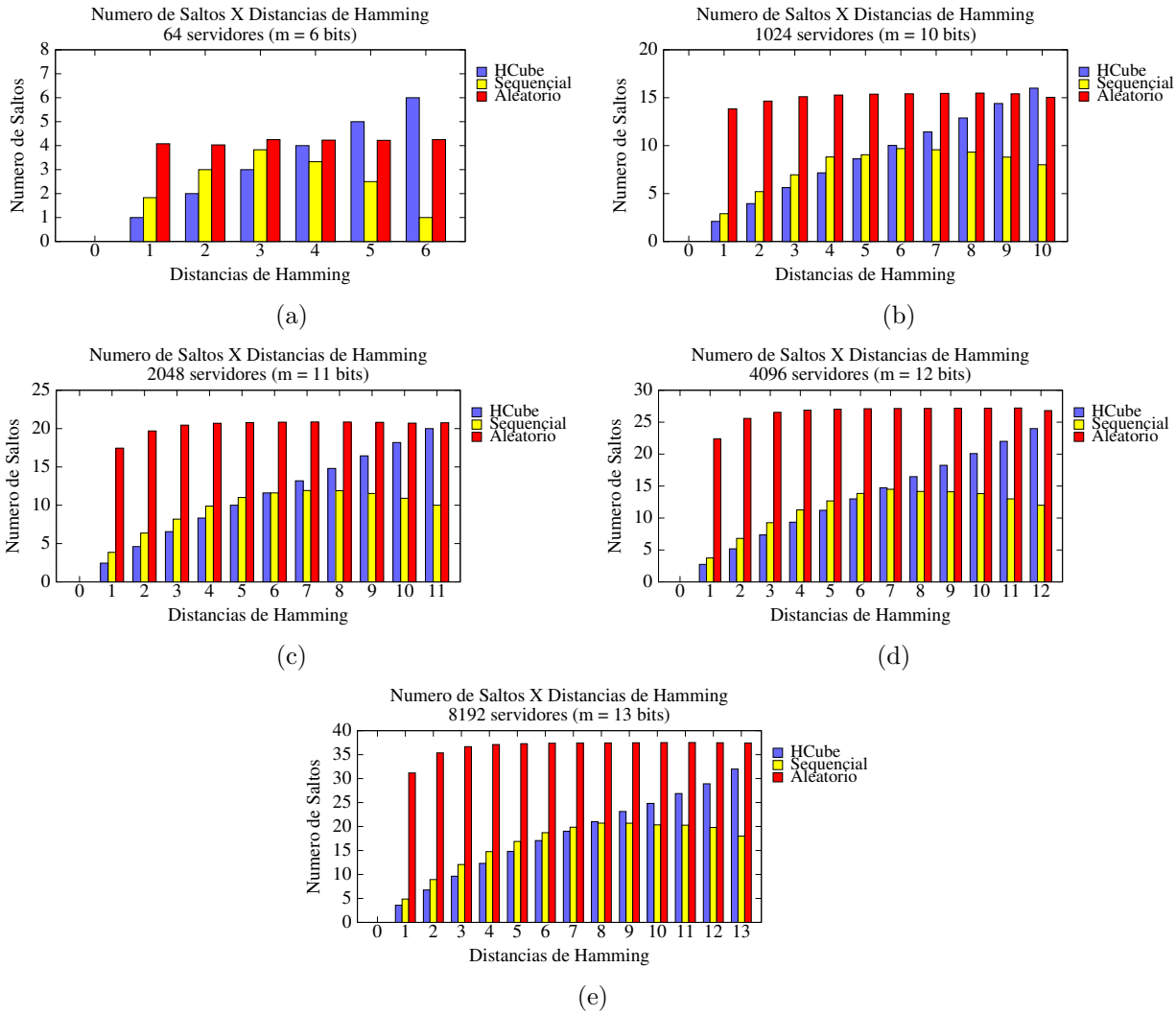


Figura 4.5: Distância em saltos X Distância de Hamming entre os servidores nas 3 diferentes organizações avaliadas. DCs de diferentes tamanhos: 64 servidores (a), 1024 servidores (b), 2048 servidores (c), 4096 servidores (d) e 8192 servidores (e).

Tabela 4.3: Média do número de saltos no HCube (organização segundo a curva de Gray).

Similaridade	$m = 6$	$m = 10$	$m = 11$	$m = 12$	$m = 13$
≥ 0.7	1.08	3.09	4.34	4.69	7.64
≥ 0.8	0.98	2.56	3.45	3.71	6.27
≥ 0.9	0.74	1.76	2,33	2.57	3.48
≥ 0.95	0.18	0.28	0.76	1.02	2.10

Tabela 4.4: Média do número de saltos no DC com organização sequencial.

Similaridade	$m = 6$	$m = 10$	$m = 11$	$m = 12$	$m = 13$
≥ 0.7	1.89	3.75	6.16	8.63	10.61
≥ 0.8	1.73	4.89	5.32	7,60	9.69
≥ 0.9	1.35	3.19	5.08	6.17	8.58
≥ 0.95	0.85	1.35	2.43	3.12	5.11

Tabela 4.5: Média do número de saltos no DC com organização aleatória.

Similaridade	$m = 6$	$m = 10$	$m = 11$	$m = 12$	$m = 13$
≥ 0.7	2.83	12.10	17.91	22.84	29.75
≥ 0.8	2.49	10.69	16.74	21.02	26.67
≥ 0.9	1.86	8.50	16.16	17.93	20.20
≥ 0.95	1.24	4.98	6.42	7.93	9.00

Conforme dito anteriormente, avaliou-se 10% do total de pares de perfis similares em cada nível de similaridade, selecionados aleatoriamente e avaliados com intervalo de confiança de 95%. Os resultados mostram que, no HCube, quanto maior o nível de similaridade desejado em uma consulta, menor é a distância entre conteúdos similares. Além disso, as menores distâncias entre conteúdos similares são obtidas com a organização proposta do HCube, de acordo com a sequência da curva de Gray.

O HCube com 64 servidores é um cubo de Hamming completo. No caso de um cubo de Hamming completo com n dimensões, dado um nível de similaridade de Hamming sim_h entre os identificadores de servidores, a distância entre eles é n - (sim_h*n). Por exemplo: com $n = 10$ e $sim_h = 0.8$, a distância entre servidores com esse nível de similaridade é 10 - (0.8 * 10) = 2. Um cubo com com mais de 64 servidores é dito incompleto e apresenta valores médios maiores do que o valor máximo obtido em um cubo completo devido ao limite de interfaces entre os servidores. Por exemplo, para 64 servidores e similaridade 0.8 o HCube apresenta uma média de 0.98 saltos e um máximo de 2 saltos. Para 1024 servidores temos valores maiores do que 2 saltos com média 2.56.

4.2.4 Cobertura das buscas por similaridade

Nesse teste mapeia-se o servidor de hospedagem (si) responsável por um determinado conteúdo indicado na mensagem $get(k,v)$, lembrando que os conteúdos similares ao conteúdo k indicado na mensagem e contidos nesse servidor serão contabilizados com uma distância de 0 saltos. A partir desse servidor, expande-se a busca para os vizinhos e os vizinhos dos vizinhos do servidor de hospedagem. A profundidade das buscas por similaridade foi expandida indefinidamente até que se completasse 100% de cobertura, uma vez que o objetivo era medir a cobertura em função da profundidade (representada pela distância em saltos). Cabe ressaltar aqui a mesma observação feita para os testes da Hamming DHT: uma vez que a base de dados é previamente conhecida, sabe-se com antecedência quais são os conteúdos similares a uma determinada referência. As Figuras 4.6, 4.7, 4.8, 4.9 e 4.10 trazem os resultados da cobertura média obtida nas buscas por similaridade nas três diferentes organizações do DC, (a) HCube (Gray), (b) Sequencial, (c) Aleatório. Foram avaliadas buscas com similaridades 0.7, 0.8, 0.9 e 0.95 em DCs com 64, 1024, 2048, 4096 e 8192 servidores.

A partir da figura pode-se observar que os melhores resultados são obtidos para os níveis de similaridade mais elevados. Por exemplo, considerando um HCube com 8192 servidores e

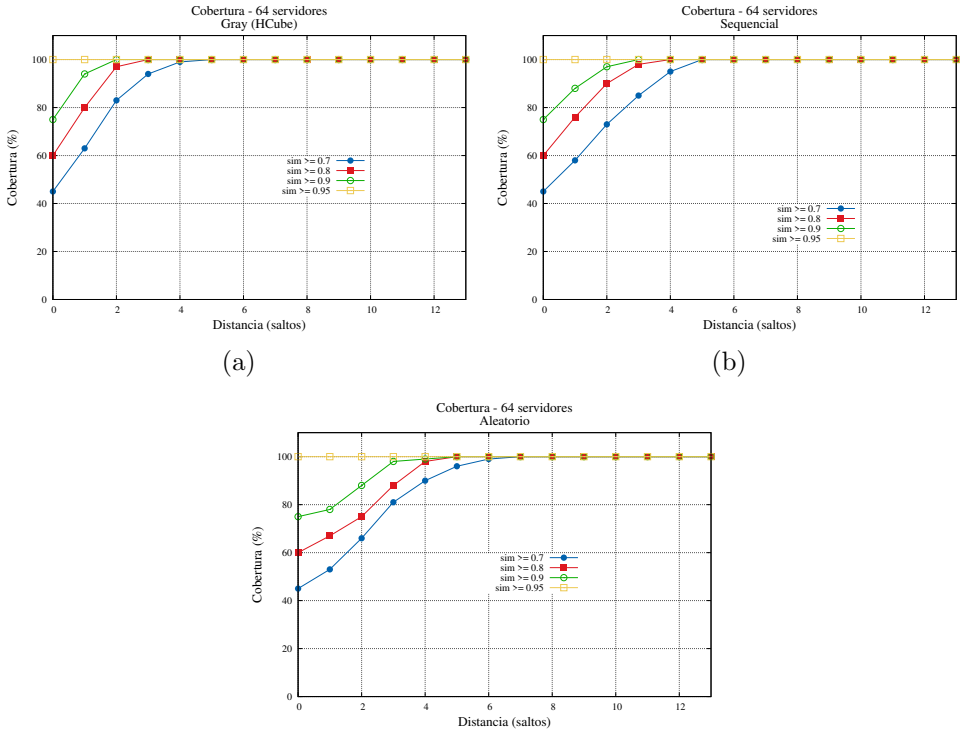


Figura 4.6: Cobertura por níveis de similaridade em um DC com 64 servidores. (a) HCube (Gray), (b) Sequencial, (c) Aleatório.

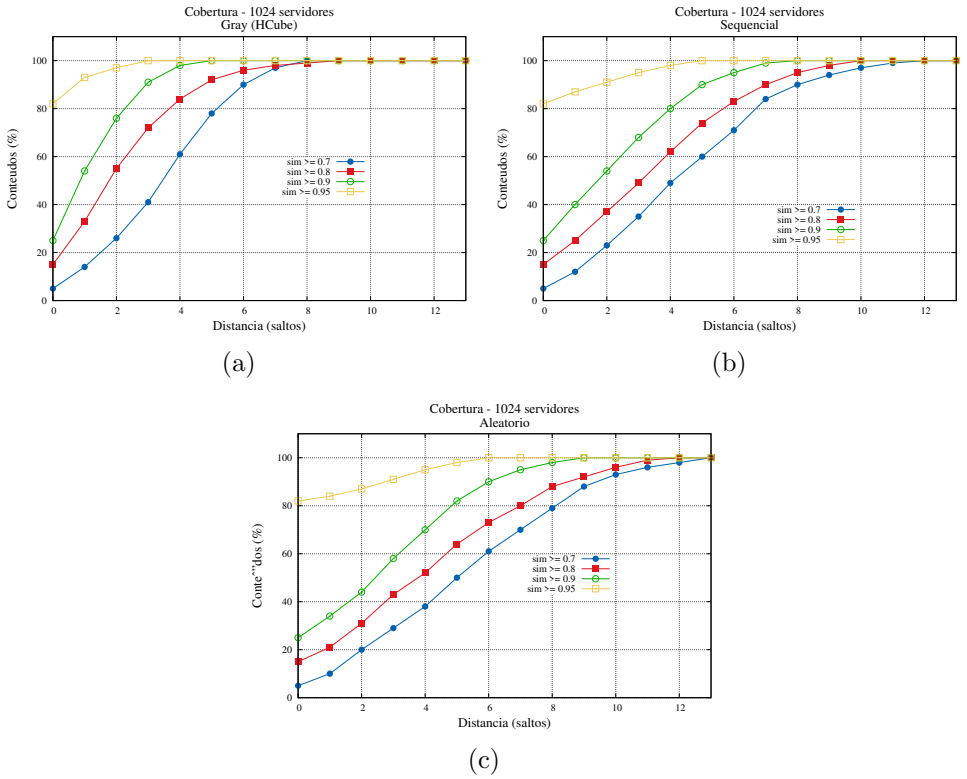


Figura 4.7: Cobertura por níveis de similaridade em um DC com 1024 servidores. (a) HCube (Gray), (b) Sequencial, (c) Aleatório.

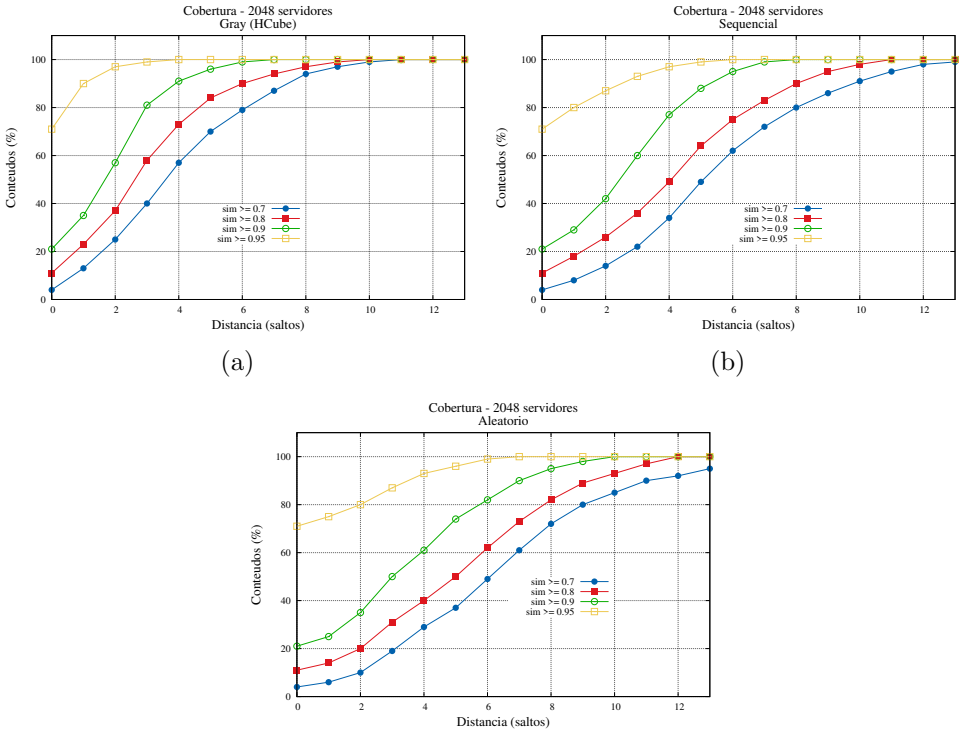


Figura 4.8: Cobertura por níveis de similaridade em um DC com 2048 servidores. (a) HCube (Gray), (b) Sequencial, (c) Aleatório.

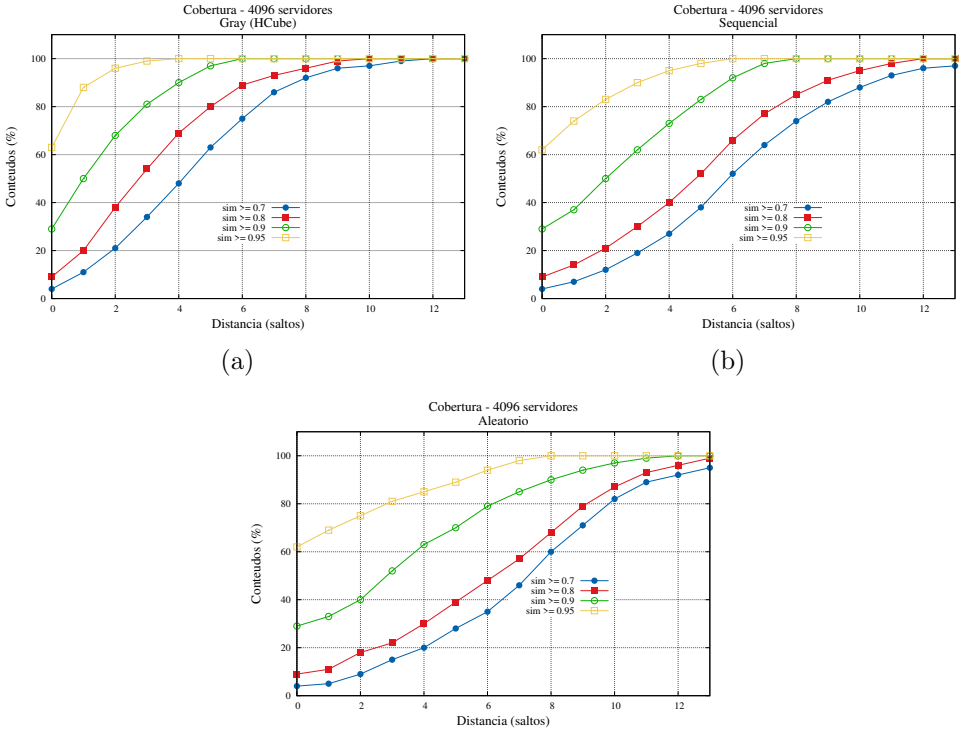


Figura 4.9: Cobertura por níveis de similaridade em um DC com 4096 servidores. (a) HCube (Gray), (b) Sequencial, (c) Aleatório.

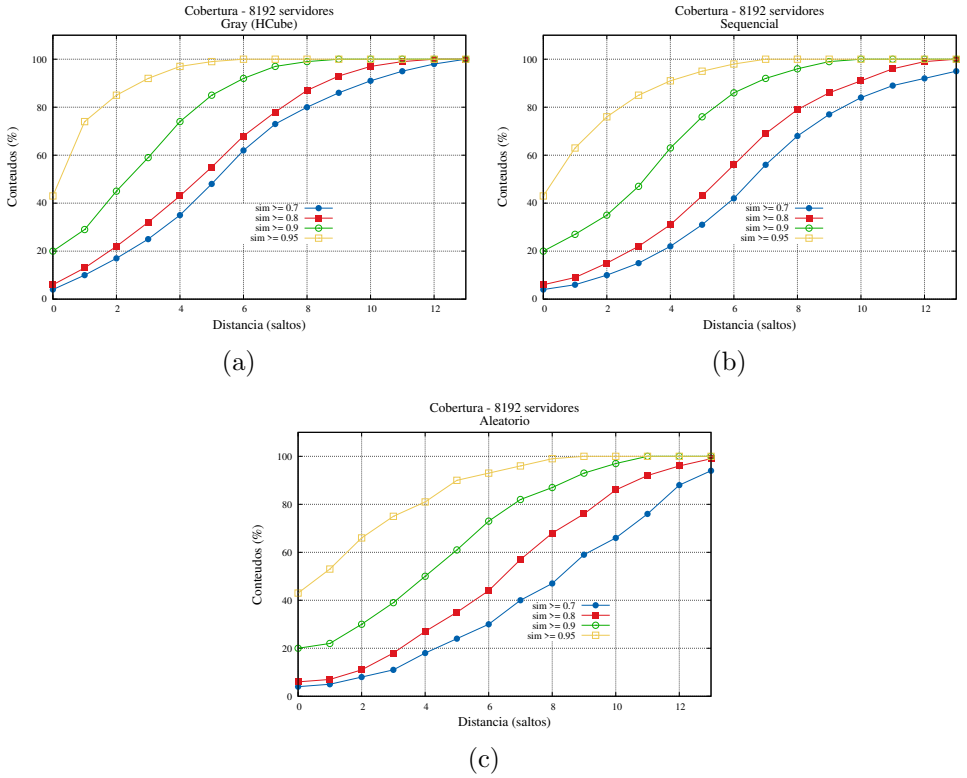


Figura 4.10: Cobertura por níveis de similaridade em um DC com 8192 servidores. (a) HCube (Gray), (b) Sequencial, (c) Aleatório.

nível de similaridade 0,9 (Figura 4.10(a)), 40% dos perfis similares são recuperados utilizando-se uma média de 2 saltos e 70% deles são recuperados com 4 saltos em média. Com o mesmo tamanho de DC, o mesmo nível de similaridade e a mesma profundidade nas buscas, 2 e 4 saltos, na organização sequencial obtém-se uma cobertura média de 30% e 62% (Figura 4.10(b)) e na organização aleatória, 9% e 11% (Figura 4.10(c)). Na figura limitou-se a profundidade máxima em 13 saltos, apenas para efeito de visualização e comparações. Principalmente na organização aleatória observa-se que não se consegue recuperar todos os conteúdos similares com essa limitação.

Em uma implementação real, não simulada, esses dados servem como base de orientação para que o administrador do DC possa regular a profundidade da busca em função da cobertura desejada. O mais importante nesse caso é a confirmação de que o HCube apresenta um bom resultado de cobertura para uma mesma profundidade.

4.3 Considerações Finais

A maioria dos trabalhos relacionados sobre busca por similaridade disponíveis na literatura é baseada em esquemas de indexação centralizados (Zhang, Agrawal, Chen & Tung 2011), funções de *hash* (Indyk & Motwani 1998) e SFCs (Lawder 1999). Alguns outros estão relacionados a soluções distribuídas em ambientes de redes P2P (Bhattacharya et al. 2005), (Tang, Xu & Mahalingam 2003) e (Haghani et al. 2009). Ambos cenários, centralizado e distribuído, são

motivados pelo problema da busca pelos vizinhos mais próximos, ou seja: "como recuperar os dados mais similares dada uma referência e um espaço de indexação"?

As soluções centralizadas oferecem resultados eficientes usando estruturas de dados especializadas. Entretanto, elas não são adequadas no cenário de *Big Data* devido à enorme quantidade de dados a serem considerados, onde as soluções distribuídas são mais interessantes. As soluções baseadas em redes P2P sobrepostas (*overlay*) aparecem como alternativa para o tratamento de grandes volumes de dados. Em geral, essas soluções são baseadas em Tabelas *Hash* Distribuídas (DHTs, ou *Distributed Hash Tables*). No entanto, embora pares que armazenam dados similares possam ser rapidamente localizados na rede P2P, não há garantia de que eles estejam fisicamente próximos, uma vez que a rede é sobreposta, podendo resultar em alta latência na recuperação desses dados.

Comparado-se com todos esses trabalhos relacionados, o HCube é uma solução distribuída que utiliza funções LSH e SFCs para suportar buscas por similaridade porém não se baseando em solução de redes sobrepostas. O HCube apresenta um DC centrado nos servidores, especializado na busca por similaridade, onde dados similares são armazenados em servidores fisicamente próximos.

Grandes empresas como Google (Chang, Dean, Ghemawat, Hsieh, Wallach, Burrows, Chandra, Fikes & Gruber 2006) e Amazon (DeCandia, Hastorun, Jampani, Kakulapati, Lakshman, Pilchin, Sivasubramanian, Vosshall & Vogels 2007) desenvolveram DCs especializados no armazenamento e processamento de grandes volumes de dados através da solução MapReduce, mas nenhum deles é especializado na busca por similaridade. Deste modo, abre-se com o HCube um novo campo de pesquisa, no qual as aplicações podem se beneficiar da sua estrutura adaptada a esse tipo de busca.

Sabe-se que é possível implementar a busca por similaridade utilizando-se de tecnologias mais difundidas e já estabelecidas na literatura, tais como o Hadoop (The Apache Software Foundation 2013) e o MapReduce (Dean & Ghemawat 2008). Aliás, inclusive, acreditamos que é possível usar a similaridade de Hamming para recuperação de conteúdos similares em DCs em conjunto com essas duas tecnologias. Ao encaminhar as consultas para os vizinhos estamos fazendo um "Map", e ao juntar os resultados no servidor de admissão estamos fazendo um "Reduce". A proposta presente nessa tese baseia-se na junção das três ferramentas que, se utilizadas em conjunto, tornam-se bastante úteis nos cenários de busca por similaridade.

A partir dos resultados de cobertura observa-se um comportamento semelhante ao presente na Hamming DHT: os resultados se tornam melhores quando se expande a busca para os vizinhos do servidor de hospedagem si. Considerando-se somente os conteúdos presentes nesse servidor (ou seja, distância 0) praticamente não existe diferença entre as organizações de DC, visto que o mapeamento do si é feito pela função de *hash* RHH, sendo este um resultado exclusivo dessa função. Como a contagem de distância é feita sempre a partir do servidor si, e esse servidor existirá em todas as implementações, é razoável esperar que eles armazenem a mesma quantidade de conteúdos similares à distância 0. Entretanto, a organização dos enlaces entre os servidores privilegiando a similaridade de Hamming influi nos resultados com distâncias superiores a 1 (um).

Novamente sabemos que o desbalanço de cargas entre os servidores pode ser uma caracterís-

tica indesejável em DC, e que esse desbalanço é um efeito colateral comumente encontrado em sistemas distribuídos baseados em *hash* LSH. Entretanto, é também sabido que existem várias técnicas de abordagem na literatura voltadas para reduzir esse efeito do desbalanceamento de carga. Por exemplo, baseando-se no trabalho apresentado em (Haghani et al. 2009) pode-se amostrar aleatoriamente a base de dados de tal forma a estimar a distribuição de similaridade entre os conteúdos armazenados no DC e, em função dessa distribuição, adotar políticas de replicação de conteúdo voltadas a equilibrar a carga entre os servidores. Essa abordagem é dinâmica pois exige uma verificação constante dessa distribuição. Uma outra possibilidade, apresentada por (Surana, Godfrey, Lakshminarayanan, Karp & Stoica 2006) em cenários de redes sobrepostas do tipo DHT, pode ser adaptada ao HCube. Trata-se da nomeação de servidores virtuais, expandindo-se o tamanho do identificador dos servidores, e mapeamento desses servidores virtuais em servidores reais do DC de uma forma determinística. Dessa forma um conteúdo estaria sempre replicado no DC. Adotando-se uma função de mapeamento baseada no oposto da similaridade, um conteúdo poderia ser mapeado no seu respectivo si e no servidor cujo identificador é o mais distante (em Hamming) possível de si, balanceando-se naturalmente a carga no HCube.

Para melhorar o desempenho do HCube é possível utilizar-se de *cache* de conteúdo nos servidores do HCube. Espalhando-se cópias dos conteúdos no caminho entre o servidor de hospedagem si, início da busca, e o servidor de hospedagem $si'''^{\cdots}$, responsável pelo armazenamento dos conteúdos similares mais distantes, seria possível reduzir ainda mais as distâncias e aumentar a cobertura. Inclusive, sabendo-se que um DC é um ambiente mais estável e controlado do que uma DHT, pode-se analisar dinamicamente as tabelas de roteamento dos servidores, geradas pelo mecanismo do roteamento XOR, e escolher os melhores pontos para instalação desses *caches*.

Uma outra limitação dos testes foi a ausência de ocorrência de falhas entre servidores do DC. Em todos os cenários testados foram avaliados somente HCubes com número de servidores igual a uma potência de 2. Entretanto, o mecanismo direto de suporte às falhas de servidores em DCs é seguir a curva de Gray e o servidor sucessor na curva assume o armazenamento temporariamente através de esquemas de replicação de conteúdo e mecanismos de *keep-alive*.

O ambiente de testes usado no desenvolvimento desta tese foi insuficiente para que pudessem ser realizados testes com mais de 8k servidores no DC. Entretanto, já nas figuras e nos resultados é possível observar que os resultados vão gradativamente "piorando" quando se aproxima de um número alto de servidores no DC. Isso acontece devido à limitação imposta de seis interfaces por servidor. Devido a essa limitação, somente seis servidores com identificadores similares podem estar diretamente conectados. Por exemplo, em um DC com 64 servidores (2^6) todos os servidores similares com distância de Hamming igual a 1 estão diretamente conectados. Em um servidor com 256 servidores (2^8), 6 servidores similares estão diretamente conectados e 2 estão localizados em distâncias maiores do que 1 salto, o que representa $\frac{2}{8} = 25\%$ das interfaces. Com o aumento do número de servidores, essa relação aumenta ainda mais até que, no limite, os resultados aproximar-se-ão do DC aleatório. Como solução para isso pode-se adotar alguma hierarquia entre DCs ou, ainda, uma sobreposição de DCs como uma rede *overlay*, criando-se atalhos no plano lógico dos identificadores dos servidores. Esse é um possível trabalho futuro

envolvendo o HCube.

Por fim, o mapeamento dos conteúdos aos servidores no HCube poderia ser feito utilizando-se técnicas diferentes, por exemplo. Na primeira, ao invés de se aplicar novamente a função RHH no identificador do conteúdo *di* para reduzi-lo ao tamanho do identificador do servidor, *si*, como foi feito na tese, poder-se-ia escolher aleatoriamente um conjunto de *bits* em *di* que, juntos, formariam o identificador *si*. Essa técnica certamente traria um custo computacional menor, uma vez que somente operações binárias seriam necessárias na geração de *si*. Entretanto, o grau de aleatoriedade da função RHH (em função da escolha dos vetores $\overrightarrow{r}$) contribui para uma melhor distribuição dos conteúdos no HCube pois todos os *bits* do identificador *di* são considerados. Isso também dificultaria o uso malicioso dos *bits* escolhidos para sobrecarregar servidores. Na segunda técnica, os identificadores dos servidores poderiam ser gerados diretamente a partir do vetor de consulta, $\overrightarrow{q}$. Essa técnica poderia melhorar a correlação entre a similaridade do cosseno e a similaridade de Hamming para cubos menores, com 64 servidores por exemplo, uma vez que não haveria o acumulo das perdas provocadas pela aplicação em série da função RHH. Essa técnica exigiria que toda consulta fosse feita a partir do vetor de consulta, impedindo que uma busca por similaridade pudesse ser realizada diretamente a partir do identificador de conteúdo. A opção escolhida nesta tese tem essa flexibilidade.

Capítulo 5

Trabalhos Relacionados

Este capítulo apresenta alguns trabalhos relacionados encontrados na literatura científica da área. Foi deixado para o final da tese propositalmente, de tal forma que nesse momento o leitor já tenha o completo conhecimento das propostas aqui apresentadas e possa identificar as principais contribuições e as diferentes abordagens dessa proposta e dos trabalhos relacionados citados nesse capítulo. Esses trabalhos foram divididos em três categorias, listadas a seguir, sendo que cada uma compõe uma seção.

- Busca por similaridade: trabalhos envolvendo busca por similaridade, representação vetorial e funções de *hash*;
- Busca por similaridade em redes P2P: trabalhos envolvendo busca por similaridade em redes P2P do tipo DHT;
- Busca por similaridade em *Data Centers*: trabalhos envolvendo busca por similaridade em *Data Centers* e Computação em Nuvem.

5.1 Busca por similaridade

A busca por similaridade é um assunto que interessa à comunidade científica de computação há vários anos. Em (Garg & Gotlieb 1986) são apresentadas metodologias para construção de funções de *hash* capazes de preservar a ordem dos dados na geração dos identificadores, isto é, $H(K_1) \geq H(K_2)$ para todo $K_1 > K_2$, onde, na notação usada no artigo original, H é uma função de *hash* e K é uma chave de dados usada na geração dos identificadores. As propostas são aplicadas em conjuntos de dados reais extraídos de listas telefônicas da época. O artigo, em resumo, apresenta o conceito de *Locality Preserving Hashing*, ou *Order-Preserving Linear Hashing*. A proposta não considera a busca por similaridade exatamente da forma como consideramos nessa tese, entretanto, também é um tipo de busca relevante em estruturas locais de indexação, especialmente discos rígidos, objetivo do trabalho na época de seu desenvolvimento.

Um outro tipo de busca por similaridade importante na literatura são as buscas por intervalos, ou *Range Queries*, cujo conceito foi mencionado na Seção 1.1.3. Em (Faloutsos 1988) é apresentado uma técnica de criação de identificadores *hash* para conteúdos multidimensionais

capaz de facilitar a busca por intervalos. Na proposta, o identificador gerado é formado pela concatenação dos valores *hash* de cada atributo do objeto representado. Para gerar o identificador *hash* de cada atributo, os mesmos são ordenados segundo a sequência do código de Gray de acordo com os intervalos de interesse nas buscas. Como conclusão, o artigo mostra que o código de Gray oferece uma agregação de conteúdos similares segundo a distância de Hamming que é sempre maior ou igual à agregação obtida utilizando-se a ordenação natural. Por exemplo, na Figura 3.1 (no canto direito da figura) todas as ocorrências de identificadores do tipo *10** estão consecutivamente posicionadas e armazenadas no nó 30 quando se utiliza a ordenação segundo o código de Gray. Ao usar a ordenação natural, as ocorrências dividem-se nos nós 13 e 30. Esse resultado foi um dos principais inspiradores desta tese.

(Indyk & Motwani 1998) e (Gionis et al. 1999) são referências fundamentais em qualquer trabalho que aborde a busca por similaridade através de funções de *hash*. Ambos apresentam o conceito de funções de *hash* sensíveis à localidade, ou *Locality Sensitive Hashing*, onde pontos em um espaço de representação multidimensional são projetados em um espaço de dimensões reduzidas de tal forma que pontos próximos no espaço de maior dimensão são mapeados no mesmo ponto, ou em pontos próximos, do espaço de menor dimensão, em função do grau de similaridade entre esses pontos. Essa medida de similaridade representa uma esfera n-dimensional no espaço, equivalente a um raio de vizinhança em torno de um ponto no espaço. O trabalho desenvolvido por (Indyk & Motwani 1998) e (Gionis et al. 1999) foi motivado pela maldição da dimensionalidade, onde as técnicas até então utilizadas para indexação visando a busca por similaridade estavam tornando-se ineficientes e, conforme já fora dito anteriormente, eram geralmente baseadas nas curvas de preenchimento de espaço. (Jagadish et al. 1995) é um bom exemplo de trabalho nesse contexto.

Em (Charikar 2002), Charikar apresentou ao mundo científico a função *Random Hyperplane Hashing* (RHH), onde os identificadores *hash* gerados pela sua aplicação em pontos de um espaço multidimensional seriam mapeados em um hipercubo de Hamming. Vértices vizinhos neste hipercubo apresentam distância de Hamming igual a 1. Pontos próximos neste espaço multidimensional seriam mapeados no mesmo vértice ou em vértices vizinhos do hipercubo de Hamming. Além disso, define-se o cosseno como medida de similaridade entre os pontos do espaço original. Como consequência, os identificadores de objetos representados por vetores multidimensionais, gerados usando-se a função RHH, apresentariam baixas distâncias de Hamming entre si, proporcionais à similaridade do cosseno entre os vetores de objetos.

Em (de Paula 2011), trabalho desenvolvido pelo grupo de pesquisa no qual essa tese está inserida, mostrou-se como essa técnica de *hash* pode ser aplicada para geração de identificadores voltados para a busca conceitual. Nela, os objetos são representados por conceitos mapeados em ontologias. Na busca conceitual o objetivo é classificar e permitir a recuperação de objetos relacionados a conceitos em uma ontologia. Admite-se também um grau de similaridade entre os conceitos medidos em função das caraterísticas da ontologia utilizada. As primeiras versões de DHT desenvolvidas durante esta tese foram apresentadas em (de Paula et al. 2010a) e (Villaça, de Paula, Magalhães & Verdi 2010) e, posteriormente, buscou-se aplicar as ideias em cenários de busca por similaridade de conteúdos, mais comuns na literatura. A partir desse ponto deu-se início ao desenvolvimento da Hamming DHT.

A representação vetorial de objetos é uma técnica bastante comum e muito utilizada em diversos trabalhos na literatura. Em (Berry et al. 1999) é apresentado de forma bastante clara e didática a fundamentação matemática necessária para a aplicação dos conceitos de Álgebra Linear no gerenciamento e indexação de grandes volumes de textos, denominada *Vector Space Model* (VSM). Basicamente, mostra-se uma maneira de se representar textos de maneira vetorial através da seleção de palavras-chave, e do uso de técnicas de indexação semântica latente, ou *Latent Semantic Indexing* (LSI) (Deerwester, Dumais, Furnas, Landauer & Harshman 1990) para reduzir as dimensões dos vetores de representação. Essa redução ocorre, resumidamente, pelo agrupamento de palavras de mesma semântica em uma mesma dimensão.

A representação de imagens através de vetores também é possível e é frequentemente utilizada em diversos trabalhos. Por exemplo, em (Kong 2009) as imagens são representadas por vetores contendo informações numéricas sobre o histograma de cores, histograma de borda (*edge*) e direção. A medição de similaridade é feita usando-se medidas de distância Euclideana e cosseno combinando-as nas operações de busca por similaridade. Esse trabalho está inserido no contexto da busca de imagens baseadas em seu conteúdo, ou *Content-Based Image Retrieval* (CBIR) onde há a extração automática de características das imagens, tais como cor, textura, formas, *layouts*, etc, com o objetivo de representá-las através de vetores em um espaço multidimensional. (Grauman & Darrell 2007) é outro trabalho motivado pela recuperação de imagens por similaridade. Entretanto, em sua proposta, Grauman apresenta uma técnica de hash LSH específica para aplicação em imagens de tal forma a beneficiar a busca por similaridade. A técnica proposta baseia-se na função RHH e, consequentemente, também apresenta a similaridade na distância de Hamming.

O desenvolvimento desta tese voltou-se para uma solução de representação e *hash* de conteúdos que fosse genérica e não focasse em uma aplicação específica. Entretanto, toda a infraestrutura proposta pode ser perfeitamente aplicada em cenários específicos a um determinado tipo de conteúdo que possa ser representado por vetores e apresente identificadores similares na distância de Hamming, inclusive o trabalho desenvolvido em (Grauman & Darrell 2007).

Áudio e vídeo também podem utilizar-se de funções LSH e da representação vetorial em cenários de busca por similaridade. Em (Ryynanen & Klapuri 2008) o usuário do sistema proposto indica um trecho de uma música ou som que se queira recuperar e todas as músicas indexadas na base de dados que possuam trechos similares são retornadas. O método utilizado automaticamente converte a consulta em notas (frequências) musicais equivalentes no formato MIDI (*Musical Instrument Digital Interface*) formando-se um vetor de alturas tonais, ou *pitches*. Em (Yoshida & Murabayashi 2008) utiliza-se funções LSH e representação vetorial para identificar vídeos similares em uma base de dados com o objetivo de identificar, em uma grande base de dados, vídeos que infrinjam termos de *copyright*, comparando os seus conteúdos com o vídeo original. Trata-se de um problema de busca por similaridade e os autores usam o conceito de "assinatura RGB (*Red, Green, Blue*)" para construir a representação vetorial dos vídeos analisados.

Entretanto, como já fora dito anteriormente, em todos os casos esses objetos passam por algum processo de extração de atributos que são posteriormente transformados em números e, a partir de então, são representados por vetores. No caso de dados que possuem atributos

não numéricos para serem analisados, o uso da representação vetorial e da indexação usando LSH precisa de algum tratamento especial. Em (Le & Ho 2005) são apresentadas diferentes técnicas de medidas de similaridade para dados categóricos, ou seja, dados não numéricos, onde os atributos representam categorias segundo alguma classificação prévia. Os resultados obtidos nesse trabalho indicam que nenhum método é dominante sobre os demais. Além disso, em (Desai et al. 2011), afirma-se que o procedimento de transformação de dados categóricos em dados numéricos é necessário pois a noção de distância ou similaridade para atributos textuais, ou categóricos, não é tão direta e usual quanto é em dados numéricos.

Por fim, em (Qian et al. 2004) compara-se as medidas de similaridade através de cosseno e distância Euclideana em cenários de busca por similaridade. Como conclusão, os autores mostram que a medida do cosseno produz resultados semelhantes aos obtidos usando-se a medida de distância Euclideana além de possuir a vantagem de, naturalmente, já ser uma medida normalizada. Em (Lee et al. 2011) é abordado o problema da redução de duplicatas em busca por similaridade, conhecido como *Similarity Joins*. Nesse trabalho, os autores sugerem o uso do cosseno como medida de similaridade visto que ela é uma medida que produz bons resultados em diversos e diferentes domínios de aplicação.

5.2 Busca por similaridade em redes P2P

Em trabalho recentemente publicado, (Zhang, Xiao, Tang & Tang 2011) apresenta um apanhado de trabalhos relacionados à indexação de dados multidimensionais em ambientes distribuídos do tipo redes P2P. Nele, fez-se uma revisão bastante completa da literatura, indicando-se os principais grupos de pesquisa ativos e seus focos de atuação, dividindo-os em 4 grupos: 1) extensão dos métodos centralizados; 2) extensão de redes P2P para suporte à busca por dados multidimensionais; 3) híbridos, que combinam técnicas centralizadas e redes P2P já existentes e consagradas na literatura e; 4) miscelânea. A Hamming DHT nesse contexto encaixa-se no grupo 3, uma vez que a partir de técnicas para indexação local, tais como LSH, RHH e Gray codes, propôs-se uma variação da DHT Chord de tal forma a capacitá-la a suportar adequadamente a busca por similaridade.

pSearch (Tang et al. 2003) é uma proposta de rede P2P que usa a representação vetorial de documentos, reduzidos segundo a técnica do LSI (Deerwester et al. 1990), e a similaridade do cosseno entre esses vetores para realização de buscas por similaridade em redes P2P. O foco do trabalho é a disseminação de um dicionário distribuído de termos que permita que os documentos possam ser indexados por todos os nós participantes da rede de maneira uniforme e distribuída. Nenhuma referência à similaridade de Hamming é mencionada e a rede proposta é específica para a busca distribuída de documentos.

Em (Haghani et al. 2009) discute-se os desafios de se usar funções LSH em redes P2P e ambientes distribuídos. O trabalho explora uma característica das funções LSH que é a habilidade de se mapear em um mesmo identificador os conteúdos similares. O objetivo, nesse caso, é mapear o espaço multidimensional dos conteúdos no espaço linear dos identificadores dos nós da rede P2P, de tal forma que conteúdos similares sejam mapeados em um mesmo nó da rede com probabilidade proporcional à medida de similaridade entre eles. Abordagem

semelhante a essa foi usada no HCube, entretanto, na Hamming DHT nós e conteúdos dividem o mesmo espaço de endereçamento. Além disso, não se discute nenhuma maneira para buscar os conteúdos similares armazenados nos outros nós da rede P2P.

Bhattacharya (Bhattacharya et al. 2005) é, sem dúvida, o trabalho mais próximo à proposta da Hamming DHT. Trata-se de um trabalho que explora a representação vetorial, similaridade do cosseno e a similaridade de Hamming na realização de buscas por similaridade. O trabalho propõe a extensão de qualquer DHT (legado) para suportar a busca por similaridade através da implementação da primitiva $get(k,sim_h)$, com a mesma definição usada na Hamming DHT. Essa implementação funciona como uma camada extra implementada acima das primitivas básicas *get* e *lookup* presentes em praticamente todas as DHTs. Na proposta, dado um conteúdo de referência k, a busca é encaminhada para o nó responsável por k através da chamada à primitiva $lookup(k)$. A partir desse nó, a camada de busca por similaridade gera todos os identificadores k' que possuem similaridade de Hamming maior ou igual a sim_h. Em seguida, todos esses identificadores k' são buscados através de chamadas sucessivas à função $get(k')$ e concentrados no nó de origem das buscas para formação da resposta.

Embora seja uma técnica que possa ser portável a todo tipo de DHT por ser uma camada implementada em cima das funções básicas das DHTs, o custo de geração de todos os identificadores é alto, diretamente proporcional ao tamanho do espaço de endereçamento e inversamente proporcional ao nível de similaridade desejado. Também há o custo do grande número de mensagens $get(k')$ trafegadas na rede. Conforme visto no Capítulo 3, a Hamming DHT inspira-se no Chord porém é uma DHT especializada na busca por similaridade e, embora não tenhamos medido esses valores, pode-se afirmar com segurança que a sobrecarga de tráfego gerada na rede é menor, comparando-se a essa proposta em (Bhattacharya et al. 2005) devido ao menor número de mensagens trocadas na rede.

HyperCup (Schlosser et al. 2003) propõe uma DHT cuja topologia é baseada em hipercubos na qual a vizinhança é adquirida segundo a distância de Hamming. Na proposta, ontologias são usadas para classificar conteúdos e guiar o roteamento na rede. Os resultados mostram que a busca por similaridade torna-se eficiente, entretanto o grande desafio em redes sobrepostas do tipo DHT cuja topologia é baseada em cubos de Hamming é o alto custo de manutenção da rede em cenários de cubos incompletos - algo bastante comum em redes dinâmicas. É consenso na literatura que a estrutura sobreposta de menor custo é o anel virtual (Lua, Crowcroft, Pias, Sharma & Lim 2005).

HyCube (Olszak 2010) é uma outra proposta de DHT inspirada no hipercubo de Hamming. Entretanto, para reduzir o custo de manutenção da rede combina-se essa topologia com uma topologia em árvore. Os nós são logicamente localizados em um *torus* n-dimensional. Resumindo, o HyCube baseia-se em cubos hierárquicos e usa transformada de Steinhaus como métrica de seleção dos próximos nós durante o roteamento. A proposta não está incluída no contexto da busca por similaridade. O objetivo principal é mostrar que o custo de manutenção de DHTs baseadas em hipercubo é reduzido com a adoção da solução apresentada.

Por fim, em (Girdzijauskas 2009) propõe-se um método de construção de DHTs baseado no fenômeno *Small World* (Kleinberg 2000). Nessa proposta o espaço de endereçamento virtual é organizado em anel e os nós estabelecem contatos diretamente com seus vizinhos adjacentes, à

direita, e à esquerda e alguns nós mais distantes em escala logarítmica. A construção proposta por (Girdzijauskas 2009) é destinada a redes cuja distribuição de conteúdos não é homogênea, ou seja, pode ser aplicada diretamente em cenários com funções de *hash* LSH, embora esse cenário não seja especificado na proposta original. O início desta tese, gerando a DHT apresentada em (Villaça et al. 2010), baseia-se fortemente nas ideias apresentadas por (Girdzijauskas 2009), tornando-se fundamental na construção da Hamming DHT.

Antes de encerrar essa seção, é importante ressaltar que as propostas envolvendo busca por similaridade em redes P2P são muitas, fazendo com que o tópico já tenha sido muito explorado na literatura. A lista de trabalhos relacionados aqui apresentada não é exaustiva e representa apenas aqueles que mais influencia tiveram durante o desenvolvimento desta tese e efetivamente contribuíram para a proposição da Hamming DHT. O *survey* apresentado em (Zhang, Xiao, Tang & Tang 2011) é prova de que esse é um tópico muito explorado mas que ainda permite contribuições, e a Hamming DHT encontrou esse espaço, visto que possui uma abordagem única na literatura relacionada tornando-se uma das principais contribuições desta tese.

5.3 Busca por similaridade em *Data Centers*

Em se tratando de *Data Center*, a grande maioria dos trabalhos relaciona-se com o uso de MapReduce (Dean & Ghemawat 2008) e Hadoop (The Apache Software Foundation 2013). O MapReduce pode ser considerado um modelo de programação paralela desenvolvido para suportar grandes volumes de dados. O modelo MapReduce é inspirado nas funções *map* e *reduce* usadas comumente em programação funcional. Processa dados em duas fases: *Map* e *Reduce*, onde a primeira fase consiste em aplicar um processamento em um repositório de dados, enquanto que a segunda fase corresponde ao processo de agregação dos resultados e sua posterior consolidação. O Hadoop, por sua vez, foi uma das primeiras implementações do MapReduce, associada a um sistema de arquivos distribuído específico para o processamento distribuído de grandes volumes de dados.

Baseando-se no MapReduce pode-se encontrar muitos trabalhos relacionados à busca por similaridade em grandes volumes de dados presentes na literatura. Dentre eles podemos citar (Kim & Shim 2012), (Li, Ju, Peng, Yu & Wang 2011) e (Wang, Chen, Huang, Xu, Yang & Xiao 2011b). Conforme já dito anteriormente, a proposta do HCube apresentada nesta tese não tem o objetivo de substituir o modelo MapReduce, inclusive pode-se aplicar esse modelo no HCube nos cenários em que a agregação de conteúdos similares for benéfica e este pode ser um caminho para pesquisas futuras na área e evolução do HCube.

A estrutura topológica do HCube foi inspirada em um hipercubo tridimensional cujo principal modelo foi obtido em (Guo et al. 2009), uma arquitetura de rede desenvolvida especialmente para *Data Centers* modulares centrados nos servidores, onde eles participam não somente do processamento mas também das funções de rede e encaminhamento de mensagens internas ao *Data Center*. Os resultados apresentados no BCube foram suficientes para torná-lo uma referência relevante para a proposta do HCube.

Além disso, em (Pasquini et al. 2011) a proposta de uso do *Data Center* inspirado no BCube foi apresentada, porém com o foco voltado para os benefícios de se trabalhar com um modelo

centrado em servidores modulares de baixo custo. Destacou-se nesse trabalho as facilidades de implementação, manutenção, custo e resiliência. Esses resultados, aliados ao BCube, foram fundamentais para a adoção do modelo de *torus* tridimensional usado no HCube.

Com relação ao roteamento XOR adotado no HCube, embora alternativas fossem possíveis (Demaine & Srinivas 1996) e (Sancho, Robles, Lopez, Flich & Duato 2003) - desde que voltadas para o roteamento por identificadores planos - a aderência natural da solução XOR, seus resultados e a experiência anteriormente obtida foi fundamental para a sua escolha e implementação no HCube. Resultados detalhados sobre o roteamento XOR no HCube são apresentados em (Villaça, Pasquini, de Paula & Magalhães 2013b).

Capítulo 6

Conclusão

A informação é um bem preciosíssimo nos dias atuais. O universo digital cresce a uma taxa de aproximadamente 7,6 PB por dia (Gantz & Reinsel 2010) e (Gens 2013) e somente 1% desses dados são processados gerando informação e conhecimento. A principal razão para isso é que a maior parte desses dados é não estruturada, o que torna difícil o seu armazenamento e recuperação usando bases de dados relacionais, as mais comumente usadas ainda hoje pelas corporações, universidades e institutos de pesquisa. Esses dados são originados nas postagens dos usuários em redes sociais, textos escritos em *blogs* e afins, fotos e vídeos disponibilizados publicamente a partir de dispositivos móveis diretamente ligados à Internet, dados de GPS (*Global Positioning Systems*), sensores de temperatura, presença, RFIDs (*Radio Frequency Identification*), imagens de câmeras de vigilância nas ruas das grandes cidades.

A existência dessa enorme quantidade de dados e as dificuldades encontradas para representação, organização, armazenamento, recuperação e processamento deram início a uma área de pesquisa que vem crescendo vertiginosamente nas áreas de Ciência da Computação, Engenharia, Estatística e Matemática: *Big Data*. É facilmente verificado na literatura relacionada que o caminho para se conseguir lidar com o *Big Data* é através de computação paralela e sistemas distribuídos. Trabalhos acadêmicos e soluções de mercado nessa área são cada vez mais frequentes e, dentre esses trabalhos e soluções, esta tese focou na busca por similaridade, um tipo de busca onde não se interessa por um conteúdo específico, mas sim por um conjunto de conteúdos que é similar a uma referência. A busca por similaridade é uma alternativa interessante no contexto de *Big Data* pois, ao invés de se buscar "uma agulha no palheiro", nos interessa buscar "agulhas em uma determinada região do palheiro", sendo esta região definida pela similaridade.

Durante o desenvolvimento desta tese procurou-se atacar dois problemas cruciais nesse cenário: a representação de dados não estruturados e o armazenamento e recuperação dos mesmos em ambientes distribuídos.

Quanto à representação de dados não estruturados essa tese abordou aspectos importantes da representação vetorial, uma das mais usadas na literatura. Embora não tenha feito nenhuma proposta inovadora, a tese contribuiu com uma importante revisão da literatura e com os resultados que mostram que, mesmo de maneira extremamente simples, é possível obter-se bons resultados com a representação vetorial. Destaca-se que o problema da representação de dados não estruturados é fundamental no cenário de *Big Data* e que a representação vetorial é uma

ferramenta importante, entretanto esse não era o foco principal da tese.

Em se tratando do armazenamento e recuperação de dados similares esta tese fez duas contribuições relevantes: a Hamming DHT e o HCube, duas estruturas distribuídas voltadas para o armazenamento e recuperação de dados através da busca por similaridade, sendo a primeira no contexto de redes sobrepostas, P2P, e o segundo no contexto de *Data Centers* e redes locais. Ambas as soluções são totalmente aderentes ao cenário de *Big Data* em ambientes distribuídos e foram o foco principal deste trabalho.

Os resultados das avaliações da Hamming DHT e do HCube mostraram que ambas as soluções atingiram o objetivo proposto: reduzir a distância entre conteúdos similares em ambientes distribuídos. A consequência dessa redução traduz-se no aumento da cobertura em buscas por similaridade. Enquanto a Hamming DHT reduz essa distância no espaço virtual, lógico, o HCube faz o mesmo serviço no espaço físico, contido nos limites da infraestrutura de rede de um *Data Center*. Ambas as soluções são importantes pois a solução sobreposta é mais livre e fácil de ser gerenciada, podendo ser usada por dados distribuídos em diferentes áreas geográficas, enquanto a solução de *Data Center* apesar de ter um custo financeiro maior devido à necessidade de se construir a infraestrutura, a redução física traduz-se na redução da latência na recuperação de dados similares, característica importante em diversas aplicações dos dias atuais.

Trabalhos Futuros

Assim como toda solução voltada para a busca por similaridade, é sabido que tanto a Hamming DHT quanto o HCube sofrem com o natural desbalanceamento de carga provocado pela agregação de conteúdo. Entretanto, assim como é certo que a adoção de uma função LSH para geração de índices de conteúdo em um sistema de armazenamento de dados implica num desbalanceamento natural da carga nesse sistema, é grande a quantidade de trabalhos relacionados dedicados a solucionar ou amenizar esse problema. Soluções de replicação de dados, *caching*, criação de nós (ou servidores) virtuais em diferentes regiões do espaço de endereçamento são possíveis. Em casos específicos, caso se conheça previamente a distribuição de conteúdos que serão armazenados, pode-se adotar estratégias explícitas de armazenamento de tal forma a reduzir o desbalanceamento. Todas essas possíveis soluções serão objetos de estudos futuros relacionados à Hamming DHT e ao HCube.

Especificamente em se tratando da Hamming DHT, o aprofundamento dos estudos voltados à geração de códigos de Gray balanceados é também uma alternativa importante para se tentar melhorar os resultados obtidos nesta tese. Sabe-se que o código de Gray binário refletido é, sem dúvida, a forma mais simples de se gerar uma sequência de Gray, e que todas as demais alternativas possuem um maior custo computacional. Em parceria com matemáticos da área de otimização combinatória, pretendemos investigar esses códigos com o objetivo de analisar o impacto deles, por exemplo, no balanceamento das infraestruturas distribuídas voltadas para a busca por similaridade contribuindo, dessa forma, para uma melhor distribuição da carga na rede. Obviamente, essa melhor distribuição da carga será acompanhada por um aumento no número de saltos médios para a busca por similaridade.

Além disso, ainda referindo-se à Hamming DHT, é sabido que é parte crucial na proposta de

redes sobrepostas do tipo P2P a avaliação do comportamento da rede em cenários dinâmicos, com uma taxa realista de entrada e saída de nós. Esse cenário exige a implementação de técnicas de monitoramento de vizinhos e replicação de carga entre os nós. A Hamming DHT herda naturalmente todos os algoritmos e soluções usadas pelo Chord. Entretanto, sabe-se que a validação da mesma nesses cenários é necessária para que a mesma seja efetivamente reconhecida na literatura. Desta forma a avaliação da Hamming DHT em cenários dinâmicos de busca por similaridade com o natural desbalanceamento de carga da rede é um desafio a ser enfrentado como trabalho futuro advindo desta tese.

Atualmente oriento um aluno de IC na Universidade Federal do ES cujo trabalho tem como objetivo extrair dados dos perfis de usuários de redes sociais, no caso o Facebook®, e a partir da similaridade entre os perfis, orientar a formação dos *fingers* em uma rede P2P. Parte-se da premissa de que usuários com perfis similares armazenam e distribuem conteúdos similares.

No HCube, a adoção de *caching* nos servidores pode ajudar a reduzir as distâncias entre os conteúdos similares. É conhecida a limitação de seis interfaces atualmente existente no HCube e sabe-se que essa limitação implica em limitações de desempenho do mesmo em Data Center de grandes proporções. O uso de *caching*, tal como usado em *Content Centric Networks* (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009), pode acelerar a busca por conteúdos similares e, uma vez que o HCube é um ambiente estável (salvo por falhas de hardware) e o algoritmo de roteamento XOR é conhecido, pode-se aplicar técnicas de otimização para determinar o melhor posicionamento do *cache* de tal forma a beneficiar a recuperação da maior quantidade de conteúdos similares que estão a mais de 1 salto do servidor de hospedagem.

O uso de servidores com mais de seis interfaces é também uma realidade atualmente. Placas de hardware com dez interfaces são encontradas no mercado e podem ajudar a melhorar o desempenho do HCube em DCs de grandes proporções. O desafio, no entanto, é melhorar a implementação usada nesta tese de tal forma a suportar a execução de mais de 8k servidores em máquinas virtuais com o hardware disponível.

Hierarquias de DCs é também um cenário a ser investigado. Se as redes sobrepostas escalam a uma quantidade enorme de nós com um baixo custo de gerenciamento, pode-se supor que uma proposta híbrida da Hamming DHT com o HCube ofereceria um meio termo entre a redução de saltos na distância física (e consequente redução da latência na busca por similaridade) e a escalabilidade da solução. Um primeiro cenário a ser investigado seria adotar um roteamento sobreposto de HCubes com tamanhos controlados, por exemplo, 1k servidores.

Sabe-se que a proposta centrada nos servidores em um hipercubo tridimensional traz inerentemente uma grande resiliência a falhas dos servidores, resultados apresentados em (Guo et al. 2009) e (Pasquini et al. 2011). Entretanto, com a atual disponibilidade de hardware de rede (roteadores e comutadores) que suportam o conceito de redes definidas por software (*Software Defined Networking*, ou SDN) torna-se viável a implementação de roteamento baseado em identificadores planos orientados pela similaridade de Hamming, sendo esse um "*hot topic*" a ser investigado como consequência desta tese.

Publicações Relacionadas à Tese como Autor Principal

- Villaça, R., de Paula, L., Magalhães, M. & Verdi, F. L. Avaliação de redes p2p baseadas no fenômeno small world para o compartilhamento de conteúdos similares gerados por funções lsh. Proceedings of the VI Workshop de Redes Dinâmicas e Sistemas P2P, WP2P '10, SBC, Gramado, RS.

- Villaça, R., de Paula, L. B., Pasquini, R. & Magalhães, M. F. Hamming DHT: An Indexing System for Similarity Search, Proc. of the 30th Brazilian Symposium on Computer Networks and Distributed Systems, SBRC '12, SBC, Ouro Preto, MG, Brazil.

- Villaça, R., de Paula, L. B., Pasquini, R. & Magalhães, M. F. Hamming DHT: Taming the Similarity Search, Proceedings of the 10th Annual IEEE Consumer Communications and Networking Conference, CCNC '13, IEEE Communications Society, Las Vegas, NV, USA.

- Villaça, R., Pasquini, R., de Paula, L. B. & Magalhães, M. F. HCube: A Server-centric Data Center Structure for Similarity Search, Proceedings of the 27th International Conference on Advanced Information Networking and Applications, AINA '13, IEEE Computer Society, Barcelona, Spain.

- Villaça, R., de Paula, L., Pasquini, R. & Magalhães, M. A similarity search system based on the Hamming distance of social profiles. 7th IEEE International Conference on Semantic Computing, ICSC' 13. California, USA.

- Villaça, R., Pasquini, R., de Paula, L. B. & Magalhães, M. F. (2014). Exploring the hamming distance to build distributed infrastructures for similarity searches, in F. Xhafa, P. Papajorgji, A. Barolli & L. Barolli (eds)„ Modelling and Processing for Next Generation Big Data Technologies and Applications (to appear), Modeling and Optimization in Science and Technology Series, Springer Berlin Heidelberg.

Submissões

- Villaça, R., Pasquini, R., de Paula, L. B. & Magalhães, M. F. (2014). HCube: Routing and Similarity Search in Data Centers. In: International Journal of Web and Grid Services (submited). Inderscience Publishers.

Publicações Relacionadas à Tese como Colaborador

- de Paula, L. B., Villaça, R. & Magalhães, M. F. Organização de Dados Multimídia para Busca Conceitual Baseada em Ontologias. Proceedings of the 15th Simpósio Brasileiro de Sistemas Multimídia e Web, Webmedia '10. SBC, Porto Alegre, RS, Brasil.

- de Paula, L. B., Villaça, R. & Magalhães, M. F. A Locality Sensitive Hashing Approach for Conceptual Classification. Proceedings of the 4th International Conference on Semantic Computing, ICSC '10. IEEE, Pittsburgh, PA, USA.

- de Paula, L. B., Villaça, R. & Magalhães, M. F. Analysis of Concept Similarity Methods Applied to an LSH Function. Computer Software and Applications Conference, COMPSAC '11. IEEE, Munich, Germany.

- de Paula, L., Villaça, R. & Magalhães, M. Organization of Multimedia Data for Conceptual Search based on Ontologies. Journal of the Brazilian Computer Society pp. 1–14. SBC, Porto Alegre, RS, Brasil.

- de Paula, L. B., Villaça, R., Magalhães, M. F. Uma proposta de função LSH baseada em ontologia para a indexação e recuperação de dados conceitualmente relacionados em redes P2P. In: 3º Seminário de Pesquisa em Ontologias no Brasil, 2010, Florianópolis/SC. Anais do 3º Seminário de Pesquisa em Ontologias no Brasil, 2010.

Demais Publicações

- de Paula, L. B., Villaça, R., Verdi, F. L., Magalhães, M. F. A Web Service-based Network Composition Architecture for the Next Generation Internet. In: 5th International Workshop on Next Generation Networking Middleware, 2008, Samos Island, Greece. 5th International Workshop on Next Generation Networking Middleware, 2008.

- Wong, W., Villaça, R. S., Pasquini, R., Paula, L. B., Verdi, F. L., Magalhães, M. F. A Framework for Mobility and Flat Addressing in Heterogeneous Domains. In: 25 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2007, Belém/PA. Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2007. p. 265-278.

Bibliografia

Agrawal, D., El Abbadi, A., Antony, S. & Das, S. (2010). Data management challenges in cloud computing infrastructures, *Proceedings of the 6th international conference on Databases in Networked Information Systems*, DNIS'10, Springer-Verlag, Berlin, Heidelberg, pp. 1–10.

Alber, J. & Niedermeier, R. (1998). On multi-dimensional hilbert indexings, *in* W.-L. Hsu & M.-Y. Kao (eds), *Computing and Combinatorics*, Vol. 1449 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 329–339.

Angiulli, F. & Pizzuti, C. (2002). Approximate k-Closest-Pairs with Space Filling Curves, *in* Y. Kambayashi, W. Winiwarter & M. Arikawa (eds), *Data Warehousing and Knowledge Discovery*, Vol. 2454 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 124–134.

Bamba, P., Subercaze, J., Gravier, C., Benmira, N. & Fontaine, J. (2012). The twitaholic next door.: scalable friend recommender system using a concept-sensitive hash function, *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, ACM, New York, NY, USA, pp. 2275–2278.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, *Scientific American* .

Berry, M. W., Drmac, Z., Elizabeth & Jessup, R. (1999). Matrices, Vector Spaces, and Information Retrieval, *SIAM Review* **41**: 335–362.

Bhattacharya, I., Kashyap, S. & Parthasarathy, S. (2005). Similarity Searching in Peer-to-Peer Databases, *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pp. 329–338.

Broder, A. Z., Charikar, M., Frieze, A. M. & Mitzenmacher, M. (1998). Min-wise independent permutations (extended abstract), *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, ACM, New York, NY, USA, pp. 327–336.

Buda, A. (2011). Life time of correlation between stocks prices on established and emerging markets, *ArXiv e-prints* . Provided by the SAO/NASA Astrophysics Data System.

Butz, A. (1971). Alternative algorithm for hilbert's space-filling curve, *Computers, IEEE Transactions on* **20**(4): 424–426.

Cederberg, J. N. (2001). *A Course in Modern Geometries*, second edn, Springer.

Cha, S.-H., Tappert, C. & Yoon, S. (2006). Enhancing Binary Feature Vector Similarity Measures, *Journal of Pattern Recognition Research* **1**(1): 63–77.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A. & Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data, *Proc. of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI '06, USENIX, Berkeley, CA, USA.

Charikar, M. S. (2002). Similarity Estimation Techniques from Rounding Algorithms, *STOC '02: Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, New York, NY, USA, pp. 380–388.

Choi, S.-S., Cha, S.-H. & Tappert, C. C. (2010). A Survey of Binary Similarity and Distance Measures, *Journal of Systemics, Cybernetics and Informatics* **8**(1): 43–48.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*, 2 edn, Routledge Academic.

Cover, T. & Hart, P. (2006). Nearest neighbor pattern classification, *IEEE Trans. Inf. Theor.* **13**(1): 21–27.

de Paula, L. B. (2011). *Utilização de Funções LSH para Busca Conceitual baseada em Ontologias.*, Phd. thesis, Faculdade de Engenharia Eletrica e Computação. Universidade Estadual de Campinas., Campinas, SP.

de Paula, L. B., Villaça, R. & Magalhães, M. F. (2010a). Organização de Dados Multimídia para Busca Conceitual Baseada em Ontologia, *Proceedings of the 15th Simpósio Brasileiro de Sistemas Multimídia e Web*, Webmedia '10, SBC, Porto Alegre, RS, Brazil.

de Paula, L. B., Villaça, R. S. & Magalhães, M. F. (2010b). A Locality Sensitive Hashing Approach for Conceptual Classification, *ICSC' 10: Proceedings of the 4th International Conference on Semantic Computing*, IEEE, Pittsburgh, PA, USA.

de Paula, L. B., Villaça, R. S. & Magalhães, M. F. (2011). Analysis of Concept Similarity Methods Applied to an LSH Function, *COMPSAC' 11: Computer Software and Applications Conference*, IEEE, Munich, Germany.

Dean, J. & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters, *Commun. ACM* **51**(1): 107–113.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. & Vogels, W. (2007). Dynamo: amazon's highly available key-value store, *SIGOPS Oper. Syst. Rev.* **41**(6): 205–220.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis, *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* **41**(6): 391–407.

Demaine, E. D. & Srinivas, S. (1996). A novel routing algorithm for k-ary n-cube interconnection networks, *International Journal of High Speed Computing* **08**(01): 81–92.

Desai, A., Singh, H. & Pudi, V. (2011). DISC: Data-Intensive Similarity Measure for Categorical Data, *in* J. Huang, L. Cao & J. Srivastava (eds), *Advances in Knowledge Discovery and Data Mining*, Vol. 6635 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 469–481.

Faloutsos, C. (1986). Multiattribute Hashing using Gray Codes, *SIGMOD Rec.* **15**(2): 227–238.

Faloutsos, C. (1988). Gray Codes for Partial Match and Range Queries, *IEEE Trans. Software Eng.* **14**(10): 1381–1393.

Fanning, S., Fanning, J. & Parker, S. (2013). Napster, `http://en.wikipedia.org/wiki/Napster`. [Online; Acesso em 15 de Abril de 2013].

Frank, A. & Asuncion, A. (2010). UCI machine learning repository, `http://archive.ics.uci.edu/ml`.

Frankel, J., Rubinacci, G. & Pepper, T. (2013). Gnutella, `http://en.wikipedia.org/wiki/Gnutella`. [Online; Acesso em 15 de Abril de 2013].

Gantz, J. & Reinsel, D. (2010). The Digital Universe Decade - Are You Ready?, `http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf`. [Online; Acesso em 2 de Março de 2013].

Garg, A. K. & Gotlieb, C. C. (1986). Order-preserving key transformations, *ACM Trans. Database Syst.* **11**(2): 213–234.

Gens, F. (2013). IDC Predictions 2013: Competing on the 3rd Platform, `http://www.idc.com/research/Predictions13/index.jsp#`. [Online; Acesso em 2 de Março de 2013].

Gionis, A., Indyk, P. & Motwani, R. (1999). Similarity Search in High Dimensions via Hashing, *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 518–529.

Girdzijauskas, S. (2009). *Designing Peer-to-Peer Overlays: a Small-World Perspective*, PhD thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Lausane, CH.

Grauman, K. & Darrell, T. (2007). Pyramid match hashing: Sub-linear time indexing over partial correspondences, *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, IEEE Computer Society.

Gray, F. (1953). Pulse code communications, US Patent 2 632 058.

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y. & Lu, S. (2009). Bcube: a high performance, server-centric network architecture for modular data centers, *SIGCOMM Comput. Commun. Rev.* **39**(4): 63–74.

Gupta, S. A., Gupta, A., Agrawal, D. & Abbadi, A. E. (2002). Approximate range selection queries in peer-to-peer, *In CIDR*.

Haghani, P., Michel, S. & Aberer, K. (2009). Distributed similarity search in high dimensions using locality sensitive hashing, *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, ACM, New York, NY, USA, pp. 744–755.

Hilbert, D. (1891). Ueber die stetige Abbildung einer Line auf ein Flächenstück, *Mathematische Annalen* **38**(3): 459–460.

Indyk, P. & Motwani, R. (1998). Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality, *STOC '98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, ACM, New York, NY, USA, pp. 604–613.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin del la Société Vaudoise des Sciences Naturelles* **37**: 547–579.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. & Braynard, R. L. (2009). Networking named content, *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, ACM, New York, NY, USA, pp. 1–12.

Jagadish, H. V., Mendelzon, A. O. & Milo, T. (1995). Similarity-based queries, *Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '95, ACM, New York, NY, USA, pp. 36–45.

Kaashoek, M. F. & Karger, D. R. (2003). Koorde: A simple degree-optimal distributed hash table, *in* M. F. Kaashoek & I. Stoica (eds), *Peer-to-Peer Systems II*, Vol. 2735 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 98–107.

Kagie, M., Wezel, M. v. & Groenen, P. (2009). An empirical comparison of dissimilarity measures for recommender systems, *Research Paper ERS-2009-023-MKT*, Erasmus Research Institute of Management (ERIM).
URL: *http://ideas.repec.org/p/dgr/eureri/1765015911.html*

Kim, Y. & Shim, K. (2012). Parallel top-k similarity join algorithms using mapreduce, *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pp. 510–521.

Kleinberg, J. (2000). The small-world phenomenon: an algorithm perspective, *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, ACM, New York, NY, USA, pp. 163–170.

Knuth, D. E. (1973). *The Art of Computer Programming: Sorting and Searching. Volume 3*, Addison-Wesley.

Kong, F.-H. (2009). Image retrieval using both color and texture features, *Machine Learning and Cybernetics, 2009 International Conference on*, Vol. 4, pp. 2228–2232.

Lawder, J. (1999). *The application of Space-filling Curves to the Storage and Retrieval of Multi-dimensional Data*, PhD thesis, University of London, London.

Le, S. Q. & Ho, T. B. (2005). An association-based dissimilarity measure for categorical data, *Pattern Recogn. Lett.* **26**(16): 2549–2557.

Lee, D., Park, J., Shim, J. & Lee, S. (2011). Efficient Filtering Techniques for Cosine Similarity Joins, *INFORMATION-An International Interdisciplinary Journal* **14**: 1265.

Li, R., Ju, L., Peng, Z., Yu, Z. & Wang, C. (2011). Batch text similarity search with mapreduce, *in* X. Du, W. Fan, J. Wang, Z. Peng & M. Sharaf (eds), *Web Technologies and Applications*, Vol. 6612 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 412–423.

Liao, S., Lopez, M. & Leutenegger, S. (2001). High Dimensional Similarity Search with Space Filling Curves, *Proceedings of the 17th International Conference on Data Engineering, 2001*, pp. 615–622.

Lin, D. (1998). An Information-Theoretic Definition of Similarity, *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 296–304.

Liu, A., Shen, K. & Torng, E. (2011). Large scale Hamming distance query processing, *2011 IEEE 27th International Conference on Data Engineering (ICDE)*, pp. 553–564.

Lops, P., de Gemmis, M. & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends, *in* F. Ricci, L. Rokach, B. Shapira & P. B. Kantor (eds), *Recommender Systems Handbook*, Springer, pp. 73–105.

Lua, E. K., Crowcroft, J., Pias, M., Sharma, R. & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes, *Communications Surveys Tutorials, IEEE* **7**(2): 72–93.

Morton, G. (1966). A computer oriented geodetic data base and a new technique in file sequencing, *Technical report*, IBM, Ottawa, Canada.

Nayak, R. (2011). Utilizing Past Relations and User Similarities in a Social Matching System, *Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part II*, PAKDD'11, Berlin, Heidelberg, pp. 99–110.

Olszak, A. (2010). Hycube: a dht routing system based on a hierarchical hypercube geometry, *Proc. of the 8th International Conference on Parallel Processing and Applied Mathematics*, PPAM'09, Springer-Verlag, Berlin, Heidelberg, pp. 260–269.

Park, H.-A., Park, J. H. & Lee, D. H. (2011). PKIS: Practical Keyword Index Search on Cloud Datacenter, *EURASIP J. Wireless Comm. and Networking* p. 64.

Pasquini, R. (2011). *Proposta de Roteamento Plano Baseado em uma Métrica de OU-Exclusivo e Visibilidade Local.*, Phd. thesis, Faculdade de Engenharia Eletrica e Computação. Universidade Estadual de Campinas., Campinas, SP.

Pasquini, R., Verdi, F. L. & Magalhães, M. F. (2011). Integrating servers and networking using an xor-based flat routing mechanism in 3-cube server-centric data centers, *Proc. the 29th Brazilian Symposium on Computer Networks and Distributed Systems*, SBRC '11, SBC, Campo Grande, MS, Brazil.

Peano, G. (1890). Sur une courbe, qui remplit toute une aire plane, *Mathematische Annalen* **36**: 157–160.

Petri, M. (2011). *Similarity Search Applications in Medical Images*, Phd. thesis, Ludwig-Maximilians-Universität München, Munich, Germany.

Qian, G., Sural, S., Gu, Y. & Pramanik, S. (2004). Similarity between Euclidean and Cosine Angle Distance for Nearest Neighbor Queries, *Proceedings of 2004 ACM Symposium on Applied Computing*, ACM Press, pp. 1232–1237.

Rasiwasia, N., Vasconcelos, N. & Moreno, P. (2006). Query by Semantic Example, *in* H. Sundaram, M. Naphade, J. Smith & Y. Rui (eds), *Image and Video Retrieval*, Vol. 4071 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 51–60.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. & Schenker, S. (2001). A scalable content-addressable network, *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, New York, NY, USA, pp. 161–172.

Rowstron, A. I. T. & Druschel, P. (2001). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Springer-Verlag, London, UK, pp. 329–350.

Ryynanen, M. & Klapuri, A. (2008). Query by Humming of MIDI and Audio using Locality Sensitive Hashing, *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 2249–2252.

Sancho, J., Robles, A., Lopez, P., Flich, J. & Duato, J. (2003). Routing in infiniband trade; torus network topologies, *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pp. 509–518.

Schlosser, M., Sintek, M., Decker, S. & Nejdl, W. (2003). Hypercup: hypercubes, ontologies, and efficient search on peer-to-peer networks, *Proceedings of the 1st international conference on Agents and peer-to-peer computing*, AP2PC'02, Springer-Verlag, Berlin, Heidelberg, pp. 112–124.

Smeaton, A. F., Wilkins, P., Worring, M., de Rooij, O., Chua, T.-S. & Luan, H. (2008). Content-based video retrieval: Three example systems from TRECVid, *Int. J. Imaging Syst. Technol.* **18**(2-3): 195–201.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F. & Balakrishnan, H. (2003). Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, *IEEE/ACM Trans. Netw.* **11**(1): 17–32.

Suparta, I. N. (2006). *Counting Sequences, Gray Codes and Lexicodes.*, Masters thesis, Technische Universiteit Delft., Bandung, Indonesia.

Surana, S., Godfrey, B., Lakshminarayanan, K., Karp, R. & Stoica, I. (2006). Load balancing in dynamic structured peer-to-peer systems, *Perform. Eval.* **63**(3): 217–240.

Tang, C., Xu, Z. & Mahalingam, M. (2003). psearch: information retrieval in structured overlays, *SIGCOMM Comput. Commun. Rev.* **33**: 89–94.

The Apache Software Foundation (2013). Apache® Hadoop, `http://hadoop.apache.org/`. [Online; Acesso em 5 de Março de 2013].

Tversky, A. (1977). Features of Similarity, *Psychological review* **84**(4): 327–352.

Valle, E. & Cord, M. (2009). Similarity search and indexing for high-dimensional data, *Brazilian Symposium on Databases, SBBD*, pp. 1–1.

Valle, E., Cord, M. & Philipp-Foliguet, S. (2008). High-dimensional descriptor indexing for large multimedia databases, *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, ACM, New York, NY, USA, pp. 739–748.

Villaça, R., de Paula, L. B., Pasquini, R. & Magalhães, M. F. (2012). Hamming DHT: An Indexing System for Similarity Search, *Proc. of the 30th Brazilian Symposium on Computer Networks and Distributed Systems*, SBRC '12, SBC, Ouro Preto, MG, Brazil.

Villaça, R., de Paula, L. B., Pasquini, R. & Magalhães, M. F. (2013). Hamming DHT: Taming the Similarity Search, *Proceedings of the 10th Annual IEEE Consumer Communications and Networking Conference*, CCNC '13, IEEE Communications Society, Las Vegas, NV, USA.

Villaça, R., de Paula, L., Magalhães, M. & Verdi, F. L. (2010). Avaliação de redes p2p baseadas no fenômeno small world para o compartilhamento de conteúdos similares gerados por funções lsh, *Proceedings of the VI Workshop de Redes Dinâmicas e Sistemas P2P*, WP2P '10, SBC, Gramado, RS.

Villaça, R., Pasquini, R., de Paula, L. B. & Magalhães, M. F. (2013a). HCube: A Server-centric Data Center Structure for Similarity Search, *Proceedings of the 27th International Conference on Advanced Information Networking and Applications*, AINA '13, IEEE Computer Society, Barcelona, Spain.

Villaça, R., Pasquini, R., de Paula, L. & Magalhães, M. (2013b). HCube: Routing and Similarity Search in Data Centers, *International Journal of Web and Grid Services* . submited.

Wang, Z., Chen, H., Huang, Y., Xu, Y., Yang, L. & Xiao, J. (2011a). Efficient Similarity Search on Massive Gene Data based on Cloud Computing, *2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, Vol. 2, pp. 702–706.

Wang, Z., Chen, H., Huang, Y., Xu, Y., Yang, L. & Xiao, J. (2011b). Efficient similarity search on massive gene data based on cloud computing, *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*, Vol. 2, pp. 702–706.

Yoshida, K. & Murabayashi, N. (2008). Tiny lsh for content-based copied video detection, *Applications and the Internet, 2008. SAINT 2008. International Symposium on*, pp. 89–95.

Zave, P. (2012). Using lightweight modeling to understand chord, *SIGCOMM Comput. Commun. Rev.* **42**(2): 49–57.

Zhang, C., Xiao, W., Tang, D. & Tang, J. (2011). P2P-based multidimensional indexing methods: A survey, *J. Syst. Softw.* **84**(12): 2348–2362.

Zhang, D., Agrawal, D., Chen, G. & Tung, A. K. H. (2011). Hashfile: An efficient index structure for multimedia data, *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, IEEE Computer Society, Washington, DC, USA, pp. 1103–1114.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. & Kubiatowicz, J. D. (2004). Tapestry: A resilient global-scale overlay for service deployment, *IEEE Journal on Selected Areas in Communications* **22**: 41–53.