ROGÉRIO GUARACI DOS SANTOS

SUBSTITUIÇÃO DE OBJETOS EM CACHE NA WWW BASEADO NA SEMÂNTICA DA INFORMAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

Área de Concentração: Sistemas de Computação
Orientador: Prof. Dr. Alcides Calsavara

Curitiba
2001

Pontifícia Universidade Católica do Paraná

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática Aplicada

ATA DA SESSÃO PÚBLICA DE DEFESA DE DISSERTAÇÃO DE MESTRADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

DEFESA DE DISSERTAÇÃO Nº 38

Aos 27 dias do mês de setembro de 2001 realizou-se a sessão pública de defesa da dissertação "**Substituição de Objetos em Cache na WWW Baseado na Semântica da Informação**", apresentada por **Rogério Guaraci dos Santos** como requisito parcial para a obtenção do título de **Mestre em Ciências**, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Alcides Calsavara PUCPR (Presidente)	Alcides Calsavara assinatura	APROVADO parecer
Prof. Dr. Edgard Jamhour PUCPR	Edgard Jamhour	APROVADO
Prof. Dr. Wagner Meira Jr UFMG	Wagner Meira Jr	Aprovado

Conforme as normas regimentais do PPGIA e da PUCPR, o trabalho apresentado foi considerado APROVADO SEM RESTRIÇÕES (aprovado sem restrições, aprovado com exigências ou reprovado), segundo a avaliação da maioria dos membros da Banca Examinadora acima indicada.

Observações, exigências e/ou restrições da Banca Examinadora, quando houver:

_____

_____

continuar no verso, se necessário

Prof. Dr. Carlos Alberto Maziero, Diretor
Programa de Pós-Graduação em Informática Aplicada
Pontifícia Universidade Católica do Paraná

A meus pais, MANOEL e ESTEFANIA .

Agradecimentos

Pesquisar é pisar em terreno desconhecido, inclui erro e risco, não há como fazer previsão. Como determinar um tempo para cumprir um objetivo de pesquisa? No entanto, ao fazer pesquisa, em algum momento o pesquisador deverá se render à realidade e conviver com as limitações de tempo e recurso, reconhecendo esses limites que o próprio homem impõe ao desenvolvimento da ciência. Mesmo quando há mais experimentos a fazer, mais referências a pesquisa e mais casos a verificar, em algum momento é preciso escrever algo e se contentar com aquilo, sabendo que poderia ter feito pouco melhor.

Isso também é verdade para o processo de reconhecer aqueles que tornaram o trabalho possível. Há sempre alguém esquecido e alguma forma mais eloqüente ou bonita de agradecer mas em algum momento temos de escrever sabendo dos riscos imputados pelos limites de nossa memória e habilidade literária. Dentro destas limitações, aqui vão meus agradecimentos.

Ao orientador, Alcides Calsavara, pela inestimável ajuda na realização deste trabalho. Agradeço, sinceramente, por ter indicado os caminhos a seguir, como também pela compreensão e generosidade em relação ao tempo.

Aos amigos, pela amizade, companheirismo e incentivo durante todos os momentos. Ao amigo Glauco e Kolb pela colaboração na implementação do simulador do LSR.

Aos professores, funcionários e colegas da PUC, pela convivência agradável durante os anos que passei aqui. Em especial a Manuel Marino Gonzales, Gerente Administrativo, que resolve as burocracias com dinamismo e simpatia.

A todos que direta ou indiretamente colaboraram na execução deste trabalho. Em especial a Karin Inês Janczeski pela compreensão na horas difíceis do envolvimento ao trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

"Assim fiz como se fosse um pequeno globo do mundo intelectual, tão verdadeira e fielmente quando pude descobrir".

FRANCIS BACON (1605)

Resumo

Esta dissertação introduz o algoritmo de substituição de objetos em cache na *Web* denominado *Least Semantically Related* (LSR). Em contraste com os algoritmos atualmente conhecidos, que se baseiam em propriedades físicas dos objetos, o LSR baseia-se na semântica da informação contida nos objetos: o LSR tende a favorecer a permanência no cache de objetos que possuem maior afinidade entre si, com relação à semântica da informação, eliminando do cache os objetos que tendem a ser de menos interesse para os clientes. Um algoritmo detalhado e uma estrutura de dados para o LSR são completamente projetados e implementados para fins de validação e comparação com outras estratégias de substituição de objetos em cache; são também implementadas as estratégias SIZE, LFU e LRU. Além disso, é projetado um modelo para a realização dos experimentos, incluindo toda a coleta e preparação de dados para análise. Os resultados experimentais mostram que a estratégia proposta oferece, na maioria das situações, uma medida de eficiência – probabilidade de acerto – superior às outras estratégias. As principais contribuições da dissertação são a proposta da nova estratégia de substituição de objetos em cache, a sua implementação e validação, assim como o próprio modelo e correspondentes ferramentas (chamados de filtros) de obtenção de dados para a validação. Os resultados iniciais são favoráveis à estratégia LSR e percebe-se que as perspectivas de aplicações e trabalhos futuros de pesquisa e desenvolvimento são muitas.

Palavras-chaves: algoritmos de substituição em cache, Web caching, proxy

Resumo

Esta dissertação introduz o algoritmo de substituição de objetos em cache na *Web* denominado *Least Semantically Related* (LSR). Em contraste com os algoritmos atualmente conhecidos, que se baseiam em propriedades físicas dos objetos, o LSR baseia-se na semântica da informação contida nos objetos: o LSR tende a favorecer a permanência no cache de objetos que possuem maior afinidade entre si, com relação à semântica da informação, eliminando do cache os objetos que tendem a ser de menos interesse para os clientes. Um algoritmo detalhado e uma estrutura de dados para o LSR são completamente projetados e implementados para fins de validação e comparação com outras estratégias de substituição de objetos em cache; são também implementadas as estratégias SIZE, LFU e LRU. Além disso, é projetado um modelo para a realização dos experimentos, incluindo toda a coleta e preparação de dados para análise. Os resultados experimentais mostram que a estratégia proposta oferece, na maioria das situações, uma medida de eficiência – probabilidade de acerto – superior às outras estratégias. As principais contribuições da dissertação são a proposta da nova estratégia de substituição de objetos em cache, a sua implementação e validação, assim como o próprio modelo e correspondentes ferramentas (chamados de filtros) de obtenção de dados para a validação. Os resultados iniciais são favoráveis à estratégia LSR e percebe-se que as perspectivas de aplicações e trabalhos futuros de pesquisa e desenvolvimento são muitas.

Palavras-chaves: algoritmos de substituição em cache, Web caching, proxy

Abstract

This dissertation proposes the Web cache replacement algorithm named *Least Semantically Related* (LSR). In contract with other well-known algorithms, which are based on physical properties of objects, LSR is based on the semantics of the information contained in objects: LSR tends to favor objects to stay in cache which are closer with respect to their semantics, removing from cache objects that tend to be of less interest to clients.

A detailed algorithm and a data structure for LSR are completely designed and implemented for the purpose of validation and comparison with other replacement algorithms, namely SIZE, LFU and LRU, also implemented. Besides, a framework for the experimental work is designed and verified, including the data and preparation phase. The initial experimental results show that the proposed strategy offers, in most cases, efficiency rate – the hit rate – better than other strategies.

The main contribution of this dissertation are the proposal of a new strategy to replace object in cache, its implementation and validation, and a framework and corresponding tools to obtain experimental data. The initial results favor LSR and it can be noted that many applications and future research and development works can be carried out.

Keywords: cache *replacement algorithm, Web caching, proxy*, cache.

Abstract

This dissertation proposes the Web cache replacement algorithm named *Least Semantically Related* (LSR). In contract with other well-known algorithms, which are based on physical properties of objects, LSR is based on the semantics of the information contained in objects: LSR tends to favor objects to stay in cache which are closer with respect to their semantics, removing from cache objects that tend to be of less interest to clients.

A detailed algorithm and a data structure for LSR are completely designed and implemented for the purpose of validation and comparison with other replacement algorithms, namely SIZE, LFU and LRU, also implemented. Besides, a framework for the experimental work is designed and verified, including the data and preparation phase. The initial experimental results show that the proposed strategy offers, in most cases, efficiency rate – the hit rate – better than other strategies.

The main contribution of this dissertation are the proposal of a new strategy to replace object in cache, its implementation and validation, and a framework and corresponding tools to obtain experimental data. The initial results favor LSR and it can be noted that many applications and future research and development works can be carried out.

Keywords: cache *replacement algorithm, Web caching, proxy*, cache.

CAPÍTULO 1

Introdução

Com o crescimento da utilização da *Web* e o surgimento de novas aplicações, cresce a cada dia a quantidade de dados transmitidos, tornando a infra-estrutura de comunicação existente cada vez mais sujeita à saturação. Grande parte desse tráfego é, muitas vezes, formado pela passagem de diversas cópias dos mesmos objetos de informação. Nesse contexto, a utilização de cache desempenha um papel fundamental por diversas razões. Primeiramente, porque é um meio de habilitar a *Web* a disponibilizar seus serviços dentro de níveis aceitáveis de tempo de resposta, pois esta tende a reduzir a média de latência no acesso aos objetos de informação. Em segundo lugar, a mesma permite redução no tráfego da rede em três níveis: **(i)** entre um cliente (*browser*) e um *proxy*, **(ii)** entre um cliente e um servidor, **(iii)** entre *proxies*. Em terceiro lugar, oferece redução no número de pedidos feitos aos servidores, diminuindo a chance de sobrecarregar os mesmos *(Stephen Williams et al., 1997)*.

Devido à natural limitação no tamanho de um cache (tanto em um *proxy* como em um cliente), quando este estiver com sua ocupação completa e for preciso inserir um novo objeto, um objeto (ou um conjunto de objetos) do cache deverá ser eleito para ser removido, ou seja, deverá ser feita a substituição de alguns objetos do cache por novos objetos *(Balachander Krishnamurthy et al., 1999)*. De forma abreviada, o problema consiste em responder às seguintes questões: É necessário remover objetos do cache? Se necessário, quais são os objetos a ser selecionados para remoção? Para tanto, existem os chamados algoritmos de substituição.

Observa-se na literatura que os algoritmos de substituição, atualmente utilizados na *Web* e mesmo os que estão em estudo nos meios acadêmicos, baseiam-se invariavelmente em características físicas dos objetos, tais como tamanho, tempo de validade *(Time-To-Live -- TTL)*, freqüência de acesso, custo de acesso, etc. Não se encontram, entretanto, algoritmos que explorem a semântica da informação contida nesses objetos. Para reduzir a latência de acesso, é desejável que os caches armazenem cópias de objetos relacionados a assuntos mais pessoais dos clientes *[Wessels95]*.

Nesse trabalho é introduzido o algoritmo de substituição denominado *Least Semantically Related*, ou simplesmente LSR, cujo princípio é substituir os objetos que sejam menos relacionados semanticamente com o novo objeto que entrar no cache. O algoritmo espera que os clientes tenham a tendência de procurar por novas informações que estejam relacionadas com a informação atualmente em mãos. Dessa forma, o algoritmo LSR tende a favorecer a permanência no cache de objetos que possuam maior afinidade entre si, com relação à semântica da informação, eliminando do cache os objetos que tendem a ser de menos interesse para os clientes na *Web*. As principais contribuições desse trabalho são a originalidade do princípio do LSR, uma definição formal do LSR e o detalhamento de sua implementação para fins de validação e comparação de desempenho com os algoritmos atualmente conhecidos.

1.1 Motivação

A Figura 1.1 mostra um cenário da configuração de componentes na *Web*, no qual um objeto de informação -- identificado por #1, faz a conexão direta entre o cliente1 e o servidor1. Exemplo: o cliente1 solicita um objeto de cor clara ao servidor. Primeiramente a consulta é feita no cache do servidor1, para verificar o objeto já não foi solicitado por outro cliente. Não encontrando no cache o objeto solicitado pelo cliente, o passo seguinte é consultar o banco de dados, verificando se o objeto solicitado pode ser encontrado. Neste exemplo, o objeto de cor clara está armazenado no banco de dados que posteriormente o enviará ao cache do servidor e, em seguida, ao cache do cliente1.

Pode-se ter ligação entre um cliente e um *proxy*, e entre proxies. Neste tipo de conexão, existe uma redução na média latente de acesso a objetos *(Azer Bestavros et al., 1995)*. Exemplo: o cliente2 está conectado a proxies, e este solicita um objeto de cor clara ao servidor, mas o objeto solicitado pelo cliente2 é o mesmo solicitado pelo cliente1. Neste caso, o objeto que estiver presente no cache do servidor1 será enviado ao cache do *proxy2*, que posteriormente o enviará ao cache do *proxy1* e, finalmente, ao cache do cliente2. O cliente3 faz a solicitação do objeto que é idêntico aos objetos alocados no cache dos proxies e o referido objeto retornará ao cache do cliente, não precisando, assim, acessar o banco de dados do servidor.

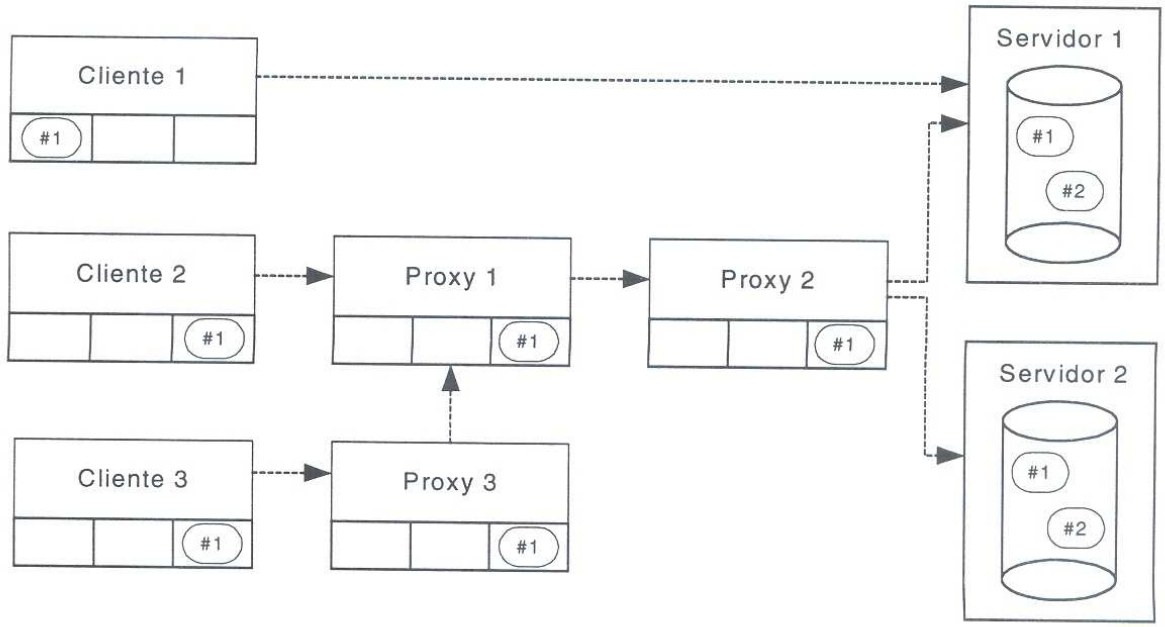


Figura 1.1 ***Cache*** **de objetos de informação em componentes da** ***Web***

1.2 Objetivo

O presente trabalho tem por objetivo definir e implementar um novo algoritmo de substituição de objetos de informação em cache, baseado na *semântica da informação*. Esse algoritmo analisará os objetos alocados em cache e separa-lo-ás de acordo com a sua semântica. O princípio do algoritmo é dar prioridade aos objetos que possuam maior relação semântica com o objeto que esteja entrando no cache, isto é, os objetos com maior relação semântica com o novo objeto tenderão a permanecer no cache.

Nesse cenário, quando um novo objeto n estiver sendo inserido no cache, outros objetos deverão ser removidos de tal forma que o espaço livre seja suficiente para a inserção do objeto n. Os objetos a serem removidos do cache serão escolhidos dentre os que possuem menor relação semântica com o objeto n, até que o espaço, ocupado pelos objetos que foram removidos, seja maior ou igual ao espaço ocupado pelo objeto n.

1.3 Contribuições

As principais contribuições do presente trabalho serão a definição de um *algoritmo de substituição baseado em semântica* e a verificação da viabilidade desse algoritmo por meio de análise estatística. Tal análise será realizada comparando o algoritmo proposto com os principais algoritmos, em uso atualmente na Internet, para a substituição de objetos em cache. Verificamos, que o algoritmo proposto é adequado para a Internet – se não de forma genérica,

mas pelo menos em algumas situações. Então, o próximo passo será, naturalmente, o emprego do algoritmo em sistemas reais.

1.4 Organização do Texto

No Capítulo 2, apresentam-se trabalhos relacionados a uma diversidade de algoritmos para substituição de objetos em cache, baseados em características físicas.

No Capítulo 3, o algoritmo LSR é introduzido e exemplificado. Apresenta-se inicialmente, o princípio de funcionamento, a fim de esclarecer como e em que o processo de desenvolvimento se apóia.

No Capítulo 4, descreve-se a implementação do algoritmo, usando um esquema baseado em árvore para a classificação de informações.

O Capítulo 5 discute sobre o experimento e validação do esquema de simulação do LSR.

O Capítulo 6 apresenta a conclusão e as propostas de trabalhos futuros.

CAPÍTULO 2

Trabalhos relacionados

Os algoritmos de substituição de *cache* têm um papel fundamental no projeto de qualquer componente de armazenamento. Esses algoritmos, por exemplo, vêm sendo intensivamente estudados no contexto da operação de sistemas de controle de memória virtual [O'Neil93 *et al.*, 1993]. Apresentam-se resumidamente, a seguir, os mais relevantes algoritmos de substituição de objetos em *cache*, baseados em características físicas dos objetos, tanto por estarem atualmente em uso na *Web*, como por estarem em estágio avançado de pesquisa. Não será feita qualquer crítica ou comparação entre estes algoritmos, pois as próprias referências bibliográficas contêm tais discussões.

- ***SIZE***: Esse algoritmo remove primeiramente do *cache*, quando da chegada de um novo objeto, o maior objeto em tamanho, de forma que possa criar espaço suficiente para o armazenamento do novo objeto, sendo que, ao remover o maior objeto em tamanho e ainda não tenha sido criado espaço suficiente para entrada no do novo objeto, deve-se retirar o próximo objeto maior até que o espaço seja suficiente. Na política *Size* podem existir objetos em igualdade no tamanho quando da escolha dos objetos para remoção; nesse caso utiliza-se a política LFU para remover dentre os objetos em igual tamanho o menos freqüentemente usado. [Aggarwal *et al.*, 1999].
- ***LEAST-RECENTLY-USED* (LRU):** Esta política privilegia as referências mais recentes, protegendo-as da substituição. Assim, ela explora o princípio da localidade temporal que diz que documentos recentemente acessados têm maior probabilidade de serem requisitados em futuro próximo. A localidade temporal foi observada em seqüência de acesso a código e dados de programas. Neste ambiente de cache tradicional, a reposição de blocos fica a cargo do algoritmo LRU por ser baseado no princípio da localidade. [Aggarwal *et al.*1999]. Há duas versões deste algoritmo:
  1. ***In-Cache*-LRU** - Nesta versão, o contador de acesso para um certo objeto será zerado (renovado) toda vez que o objeto entrar no *cache*.
  2. ***Perfect*-LRU** - Nesta versão, o contador de acesso para um certo objeto será zerado somente na primeira vez que o objeto entrar em *cache*. Se o objeto for

removido do *cache* e retornar posteriormente, o seu contador de acesso terá como valor inicial o valor que tinha na sua remoção.

- ***Least Normalized Cost Replacement*** **(LNC-R)** - Este algoritmo de substituição de objetos em *cache* maximiza o tempo de resposta. O algoritmo de substituição de *cache* LNC-R estima a probabilidade de uma futura consulta a um determinado objeto, usando uma movimentação média dos últimos tempos de chegada de K requisições do objeto. Contudo, foi observado que os clientes da *Web* têm preferência por acessar objetos pequenos. E por isso, a informação de domínio específico é baseada em estimativas da probabilidade de uma futura consulta no padrão de referência do objeto como no tamanho do objeto. O LNC-R, que é o algoritmo de menor custo de substituição de objetos normalizados, define-se :

$$DSR = Sum_i (d_i * h_i) / Sum_i (di * r_i)$$

onde d_i é a média da espera para buscar o objeto D_i para o *cache*, r_i é o número total de referências à D_i e h_i é o número de referências a D_i que foram satisfeitas a partir do *cache*. A economia de atraso determina a fração de comunicação e os atrasos do servidor, os quais serão salvos, satisfazendo as referências do *cache* no lugar das de um servidor. Se o tempo para satisfazer uma consulta ao *cache* for menor que o tempo para consultar o servidor (o que é um caso típico), a razão máxima de economia de atraso garante também o mínimo tempo de resposta à consulta do cliente.) [Peter Scheurmann *et al.*, 1997].

- ***LOWEST RELATIVE VALUE*** **(LRV)** - Este algoritmo é baseado na recenticidade, tamanho e freqüência do objeto. Os autores encontram funções matemáticas que são aproximações da distribuição dos tempos entre acessos a um mesmo objeto e da probabilidade de mais acessos dado que o objeto foi acessado previamente *i* vezes. Estas funções são baseadas em uma extensa caracterização de duas cargas de caches de rede. O cache acumula, em tempo de execução, estatísticas de cada objeto requisitado, atualizadas a cada acesso. Com base nas funções e nos valores das estatísticas, a probabilidade *Pr* de um objeto ser acessado no futuro é calculada. *Pr* é diferente para cada objeto e é dependente do tempo. Objetos com menor *Pr* são descartados. O método é dependente da função de aproximação da distribuição dos

tempos entre acessos a um mesmo objeto que, por sua vez, é baseada nas cargas estudadas. A implementação é cara. A função de aproximação utiliza as funções logarítmica e exponencial e constantes empiricamente escolhidas e é calculada freqüentemente para todos os objetos presentes no cache. A principal desvantagem do LRV, além de seu custo proibitivo para implementação, é a sua grande parametrização com base em apenas duas cargas, o que deixa dúvidas sobre o bom desempenho deste algoritmo com cargas diferentes [Rizzo *and* Vicisano 1998].

- ***LRUMIN***: Este algoritmo é uma variação do LRU, a qual verifica o tamanho do objeto a ser armazenado e procura no cache objetos de tamanho maior ou igual. Caso existam documentos que atendem a este requisito, o algoritmo LRU é aplicado a este subconjunto. Caso contrário, todos os objetos cujos tamanhos sejam maiores ou iguais à metade do tamanho do objeto a ser armazenado são incluídos no subconjunto. A operação é repetida até que o espaço necessário seja liberado. [Williams *et al.* 1996].
- **SLRU** *(Segmented* LRU): divide o cache em dois segmentos e implementa duas listas LRU, uma protegida, para objetos com mais de um acesso, e outra para objetos com apenas um acesso. Portanto considera freqüência e recenticidade. [Karedla *et al.* 1994]
- **LRU-***threshold* e **LRU**-*adaptive:* são variações de LRU nas quais há um limite para o tamanho de objetos que pode ser armazenado. Na primeira versão este limite é fixo e na segunda é alterado dinamicamente. Na versão dinâmica, a variação correspondente de desempenho indica alterações posteriores. A definição de um limite para o tamanho do objeto que pode ser armazenado é uma forma de controle de admissão e pode ser utilizado em qualquer política. O estabelecimento de um limite de tamanho para armazenamento inferior ao tamanho máximo dos objetos requisitados equivale a um corte na cauda da distribuição dos tamanhos cujo efeito é um possível aumento em HR (hit rate) acompanhado de uma diminuição do BHR (byte hit rate). [Markatos, 1996]
- ***GREEDYDUAL-SIZE (GD-SIZE):*** O algoritmo proposto é também um algoritmo híbrido que considera recenticidade dos acessos, tamanho e custo de busca de um objeto. Os autores mostram que considerar a latência não leva a bons resultados devido à grande variabilidade de latência de busca de um mesmo objeto, o que confirma resultados anteriores. O algoritmo ordena os objetos de acordo com um valor H definido por H = custo/tamanho. Objetos com menor valor H são candidatos a sair

do cache. A recenticidade é considerada da seguinte forma. A cada acesso, o valor de *H* do objeto é calculado e acumulado ao valor residual anterior. Portanto, quanto mais recente o acesso ao objeto, maior o seu *H* e menor sua probabilidade de ser retirado do cache.

A definição de custo do GD-Size depende do objetivo do algoritmo. Várias opções são analisadas, porém o melhor desempenho em HR é obtido com o custo igual a 1. Uma variação, GD-Size(pacotes), considera o número de pacotes gerados para a transmissão do objetoonde *pacotes* = 2 + *tamanho*/536[1] . Cada opção oferecida é projetada para otimizar apenas uma métrica. [Cao *and* Irani 1997].

- ***HYPER-G*** - Este algoritmo é um refinamento do algoritmo LFU, considerando ainda o tempo, de acesso e o tamanho dos objetos [Williams *et al.* 1996].
- ***PITKOW/RECKER*** - Este algoritmo substitui os objetos menos recentemente usados (LRU), exceto se todos os objetos forem acessados no mesmo dia, caso em que se remove o maior objeto [Williams *et al.* 1996].
- ***LOWEST-LATENCY-FIRST*** - Este algoritmo substitui primeiro o objeto que teve menor latência no acesso via rede. No entanto, o mesmo trabalho mostra que não é um bom critério. Em sistemas com grande variabilidade, onde picos numa medida impactam negativamente o desempenho, considerar apenas a média para representar os dados pode levar a conclusões errôneas. Portanto, a utilização destes tempos deve ser feita por meio de valores médios acompanhados por medidas de dispersão dos dados. Isto pode dificultar sua utilização. [Wooster *and* Abrams, 1998].
- ***HYBRID*** - Este algoritmo híbrido HYB que calcula, para cada objeto, seu valor para o cache. Este cálculo é feito por uma função que combina os parâmetros tempo de conexão ao servidor do objeto, *bandwidth* da conexão, a freqüência e o tamanho do objeto. Objetos com menor valor são descartados. O algoritmo apresentou bom desempenho em HR e tempo de resposta frente a LRU, LFU e SIZE. Suas desvantagens são a necessidade de armazenar mais informações para cada objeto e a necessidade de ajustes cuidadosos das constantes utilizadas na função. [Wooster *and* Abrams, 1998]. Na mesma linha está o MIX [Niclausse *et al.* 1998] que utiliza os

[1] O cálculo considera 2 pacotes para estabelecer a conexão mais o número de pacotes necessários de acordo com o tamanho do objeto, considerando que um segmento TCP transmite 536 bytes de dados

parâmetros latência, número de referências, tamanho e tempo decorrido desde a última referência.

- ***FIRST-IN, FIRST-OUT(FIFO)*** - Este algoritmo substitui o objeto que entrar primeiro no *cache* [Silberschatz *and* Galvin, 1994].
- ***KEY-BASED POLICIES*** - Este algoritmo classifica os objetos por ordem de tamanho como chave-primária e, como chave-secundária, classifica os objetos pelo tempo de acesso. Ao substituir o objeto em *cache*, caso ocorra coincid6encia de tamanho o mecanismo aplica a política do LRU para substituição [Williams *et al.*1996].
- ***FUNCTION-BASED REPLACEMENT POLICIES*** - Este mecanismo de substituição de objetos emprega uma função geral para diferentes fatores, tais como: último acesso (LFU), hora da entrada do objeto no *cache*, custo de transferência (LRV) e o tempo de validade do objeto (TTL) [Wooster *and* Abrams, 1998].
- ***PYRAMIDAL SELECTION SCHEME*** **(PSS)** - Este algoritmo possui outra política híbrida: considera o tamanho do objeto e a recenticidade do último acesso de cada objeto armazenado no cache. PSS implementa controle de admissão utilizando um cache auxiliar que contém apenas informações sobre o objeto. A freqüência dinâmica de um objeto i é definida por $1/\Delta T_{ik}$ onde ΔT_{ik} é o número de acessos ao cache desde a última vez que o objeto i foi acessado e k é a iteração atual. O objeto a ser retirado é o que tem maior $S_i * \Delta T_{ik}$ onde S_i é o tamanho de i. O objetivo da política é minimizar HR.

Para otimizar a implementação do PSS, é feita uma aproximação. Em vez de calcular o valor $S_i * \Delta T_{ik}$ para todos os objetos, estes são classificados em grupos de acordo com o tamanho e os valores são calculados apenas para o objeto menos recentemente utilizado de cada grupo. Posteriormente os escolhidos nos grupos são comparados entre si. O algoritmo apresenta bons resultados para casos de correlação positiva e para casos onde não há correlação entre o tamanho e o número de acessos. Para o caso real, negativamente correlacionado, o algoritmo não apresenta bons resultados [Aggarwal *et al.* 1996].

O desempenho de vários algoritmos em relação a HR e BHR foi comparado em [Arlitt *et al.* 1998]. Os autores analisaram políticas baseadas em freqüência (LFU e

LFU-*aging*), em recenticidade (LRU, SLRU e LRU-*k*) e em tamanho (SIZE E GD-Size). Os resultados indicaram que as políticas baseadas em tamanho obtêm melhores resultados em HR enquanto as políticas baseadas em freqüência são melhores quanto a BHR.

Em resumo, as políticas híbridas para substituição em cache na WWW ordenam os arquivos por um valor calculado através de uma fórmula do tipo [Wessels 1995]: $valor = f^a \cdot r^b \cdot t^c$, onde f, r e t correspondem, por exemplo, aos parâmetros freqüência, recenticidade e tamanho, e as constantes a, b e c são os pesos associados. Outros parâmetros, sempre identificados com os arquivos, também podem fazer parte da fórmula. Arquivos com maior *valor* são mantidos no cache. Por exemplo, a deve ser positivo, valorizando arquivos com maior freqüência e b deve ser negativo para privilegiar arquivos mais recentemente acessados. A constante c é em geral negativa, discriminando arquivos grandes. A fórmula é única para classificar todos os arquivos e obter a ordem na qual os mesmos serão removidos do cache.

- **CACHE PARTICIONADO** – divide o espaço do *cache* em partições. Cada partição é dedicada ao armazenamento dos objetos cujos tamanhos pertencem a um intervalo. Cada intervalo define uma classe de objetos. As classes são em número igual ao número de partições e cobrem todos os tamanhos possíveis de objetos, sem sobreposição.

Substituição dos objetos são feitas apenas entre objetos de uma mesma classe. Este modelo não permite nem retirada de um objeto pequeno para a inserção de um objeto grande, nem retirada de um objeto grande para a inserção de um objeto pequeno, pois não é possível fazer substituição entre objetos de tamanhos extremos. A proposta de dividir o espaço em partição que armazenem classes de objetos de tamanho similar desdobra a gerência de espaço de caches em dois campos: a organização do espaço e a política de substituição

A regra para ordenação é dada pela política de reposição. Por exemplo, a política LRU gera uma lista LRU. A política LFU gera uma lista cuja chave de ordenação é o número de acessos a cada objeto. A lista é atualizada a cada requisição. A atualização consiste na reordenação da lista em caso de *hit* ou na inserção com possibilidade de retirada em caso de *miss*. Cada inserção envolve no máximo uma retirada. Portanto o

estado do cache muda lentamente, é quase estacionário, e a correlação entre o estado do cache no passado imediato e no futuro imediato é alta. Através do particionamento foi possível provar que o desempenho em HR e BHR é função do tamanho médio dos arquivos solicitados e do tamanho médio dos *hits*. [C. Murta *et al.* 1998].

CAPÍTULO 3

O algoritmo LSR

3.1 Princípio

Quando um novo objeto *n* estiver sendo inserido no cache com tamanho maior que o espaço livre, outros objetos deverão ser removidos do cache de tal forma que o espaço livre do cache seja suficiente para a inserção do objeto *n* (John Dilley *et al.*1999). O princípio de funcionamento do algoritmo LSR é o de que os objetos a serem removidos do cache, sejam escolhidos, um de cada vez, dentre os que possuam menor relação semântica com o objeto *n*, até que o espaço, então ocupado pelos objetos removidos, seja igual ou maior que espaço ocupado pelo objeto *n*.

3.2 Pseudo-Código

A seguir descreve-se o algoritmo LSR através do pseudo-código listado na Figura 3.2. O algoritmo inicia-se com a invocação do procedimento *lsr* (linha 1), segundo o qual um objeto *n* deverá ser inserido no cache *C*, de acordo com sua semântica, removendo outros objetos se não houver espaço suficiente para *n*. Na linha 3 verifica-se se o tamanho de *n* é maior que o tamanho de *C*; se o for, aborta-se, pois não é possível inserir o objeto no cache, mesmo que este esteja completamente esvaziado. Na linha 4, a semântica *s* de *n* é extraída. Na linha 5, é definido o local *l* de *C* para a inserção de *n*, de acordo com *s*, isto é, determina-se qual é o assunto mais específico referente à informação contida em *n*. Na linha 6, verifica-se se o tamanho de *n* é maior que o espaço livre em *C*; se o for, na linha 7, cria-se espaço suficiente para que *n* caiba em *C*, removendo objetos que estejam o mais semanticamente distante de *l*. Finalmente, na linha 8, insere-se *n* no local *l*.

O procedimento de criação de espaço, invocado na linha 7, é detalhado a partir da linha 10. O conjunto *C* da Figura 3.1 denota o próprio cache, isto é, o conjunto de todos os objetos em cache. O subconjunto *D*, $D \subseteq C$, denota os objetos mais distantes semanticamente do novo objeto *n*, em um certo instante, sendo que, dentre estes, todos têm a mesma distância semântica de *n* e, por isso, todos são igualmente candidatos a remoção. O subconjunto *R*, $R \subseteq D$ contém os objetos que devem ser efetivamente removidos. Esse conjunto é formado a partir de objetos extraídos do subconjunto *D*, utilizando-se para isso um algoritmo *S* que pode

variar. Por exemplo, *S* pode ser qualquer um dos algoritmos descritos na Seção 2. Caso a remoção de todos os objetos pertencentes a *R* não criar espaço suficiente para a inserção de *n*, então um novo subconjunto *D* será definido, dando a mesma seqüência ao algoritmo. O pseudo-código, correspondente a esse princípio, situa-se entre as linhas 14 e 23, inclusive. Na linha 12, define-se que o espaço pendente de criação; vai diminuindo à medida que objetos vão sendo removidos (linha 22). Na linha 13, invoca-se um procedimento para marcar todos os assuntos mais genéricos que o assunto ao qual o novo objeto deve ser vinculado, (espaço que ainda precisa ser liberado para caber o novo objeto) representado pelo local *l*. Isto é necessário para se dar prioridade a esse ramo de assuntos na escolha de objetos para remoção. Ao final do procedimento (linha 24), os ancestrais são desmarcados com invocação do procedimento adequado.

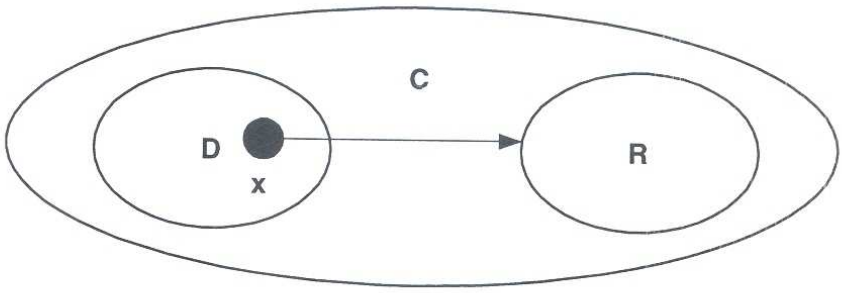


Figura 3.1 Princípios do procedimento de criação de espaço no *cache*

Todos os procedimentos invocados pelo procedimento *crie_espaço* são dependentes da estrutura utilizada para classificação de assuntos. Na Seção 4, estes procedimentos são discutidos para a implementação do algoritmo LSR, utilizando-se uma árvore para classificação dos assuntos.

```
[01] lsr ( in Cache C, in Objeto n )
     -- Insere um novo objeto n no cache C, de acordo com sua semântica e
     -- remove outros objetos, se não houver espaço suficiente para n.
[02] INÍCIO
[03]    SE (n.tamanho > C.tamanho) aborte;
[04]    s := extrai_semântica(n); -- Extrai informação semântica s de n
        -- Determina o local l do cache C para inserção do objeto n de acordo com s
[05]    l := define_local(C, s);
        -- Tamanho de n é maior que o espaço livre no cache C ?
[06]    SE (n.tamanho > C.espaço_livre)
[07]       ENTÃO cria_espaço (C, n, l); -- Cria espaço suficiente para n em C
[08]    insere(C, n, l); -- Insere n no local l do cache C
[09] FIM

[10] cria_espaço ( in Cache C, in Objeto n, in Local l )
     -- Remove objetos do cache C de acordo com o local l, tal que
     -- o espaço livre do cache seja suficiente para a inserção de n.
     -- O algoritmo de substituição S é um atributo do cache C.
[11] INÍCIO
        -- O espaço pendente de criação inicialmente é o próprio tamanho do objeto n menos
        -- o espaço atualmente disponível no cache.
[12]    p := n.tamanho - C.espaço_livre;
[13]    marque_assuntos_ancestrais( l );
[14]    REPITA
           -- Determina o conjunto D de objetos no cache C de menor relação semântica
           -- (maior distância semântica de) com n :
[15]       D := determine_objetos_mais_distantes (C, l);
[16]       R := { }; -- Inicia o conjunto R de objetos a serem removidos como vazio
[17]       REPITA
[18]          x := C.S(D); -- Aplica o algoritmo S em D, obtendo o objeto x
[19]          transfira(D ,R, x); -- Transfere x de D para R
[20]       ATÉ QUE ( vazio(D) ou R.espaço_ocupado >= p)
           -- D é vazio ou espaço ocupado por R é maior ou igual ao
           -- espaço pendente p de criação
[21]       remova(C, R) -- Remove os objetos de R do cache C
           -- Espaço pendente de criação é diminuído de acordo com os objetos removidos:
[22]       p := p - R.espaço_ocupado;
[23]    ATÉ QUE (p <= 0)--Espaço criado no cache C é suficiente para caber o objeto n ?
[24]    desmarque_assuntos_ancestrais( l );
[25] FIM
```

Figura 3.2 O algoritmo LSR

3.3 Exemplo

Os objetos alocados no cache *C*, Figura 3.3, estão logicamente agrupados de acordo com sua semântica através de conjuntos que denotam diferentes assuntos. Pode-se verificar que *esporte* é um conjunto genérico assim como *Informática*, enquanto que *Basquetebol e*

Futebol são subconjuntos de esporte; o *etc* é um subconjunto de todos os assuntos. Por exemplo, dentro de *Basquetebol* há objetos relacionados a *NBA*, Seleção brasileira e Oscar, já em Informática, os objetos relacionados são Windows e Ensino a distância. Caso seja necessário inserir um novo objeto que contenha informação sobre *Futebol* e seja necessário substituir alguns dos objetos presentes no cache, será dada preferência para a remoção de objetos relacionados com o assunto *Informática*, por estarem mais distantes semanticamente. Se o espaço livre em cache resultante dessa remoção ainda for insuficiente para inserir o novo objeto, então serão removidos objetos relacionados com *Basquetebol*. Em última instância, serão removidos objetos relacionados com *Futebol*.

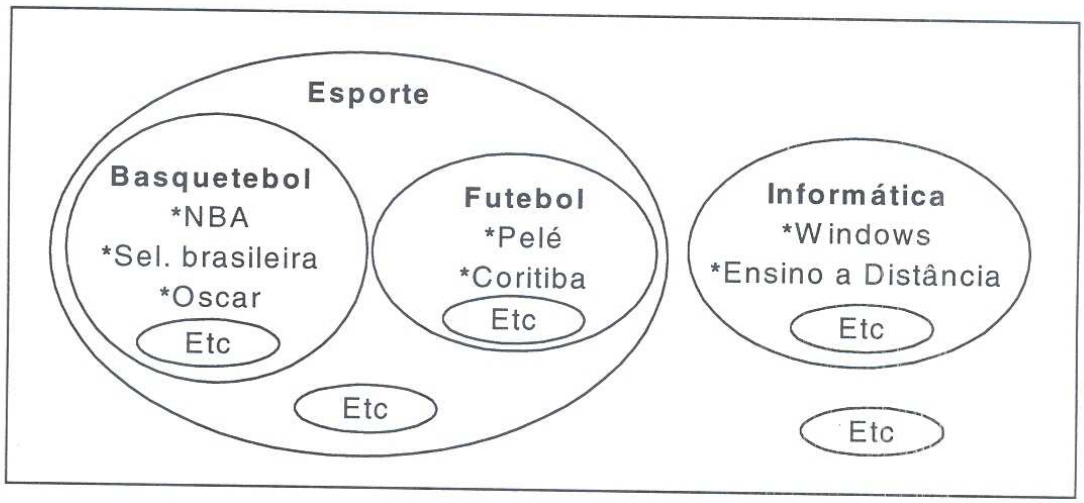


Figura 3.3 Conjunto de objetos relacionados semanticamente

CAPÍTULO 4

A implementação do LSR baseada em árvore

A classificação da informação pode ser baseada em uma hierarquia de assuntos, isto é, uma árvore cujos nós correspondam aos conjuntos e sub-conjuntos de assuntos *(Pablo Rodriguez et al., 1999)*. Cada conjunto (nó da árvore) contém $m>=0$ subconjuntos (subárvores), disjuntos S_1, S_2,..., S_m, e assim sucessivamente, e $n >= 0$ objetos de informação. Esta estrutura, em particular, permite o refinamento do LSR com relação aos procedimentos invocados pelo procedimento *crie_espaço* (Figura 3.1, linha 10).

4.1 Exemplo

A Figura 4.1 ilustra uma árvore semântica, isto é, uma árvore de assuntos com os respectivos objetos de informação vinculados. Para a representação deste exemplo, os objetos serão identificados unicamente por um número inteiro, o nó raiz da árvore, denominado *Raiz*. Esta representa o conjunto universo de assuntos e possui duas subárvores: *Esporte e Música*. O nó *Esporte* possui dois descendentes, denominados *Futebol e Basquetebol*. O nó *Música*, por sua vez, possui os descendentes *Jazz* e *Rock*. Assim, os objetos de informação podem ser vinculados à sua correspondente categoria, por exemplo, o objeto de informação #1 está vinculado à categoria *Raiz.Esporte*, enquanto que o objeto #6 está vinculado à categoria *Raiz.Música.Rock*. O nó *Etc*, presente em todas as subárvores, abriga todos os assuntos não previstos na correspondente subárvore, por exemplo, se for necessário inserir um novo objeto, cuja semântica é *Esporte.Futebol.Flamengo*, como é o caso do objeto #5, este deverá ser inserido em *Esporte.Futebol.Etc*, pois o assunto *Flamengo* não consta na subárvore *Esporte.Futebol*.

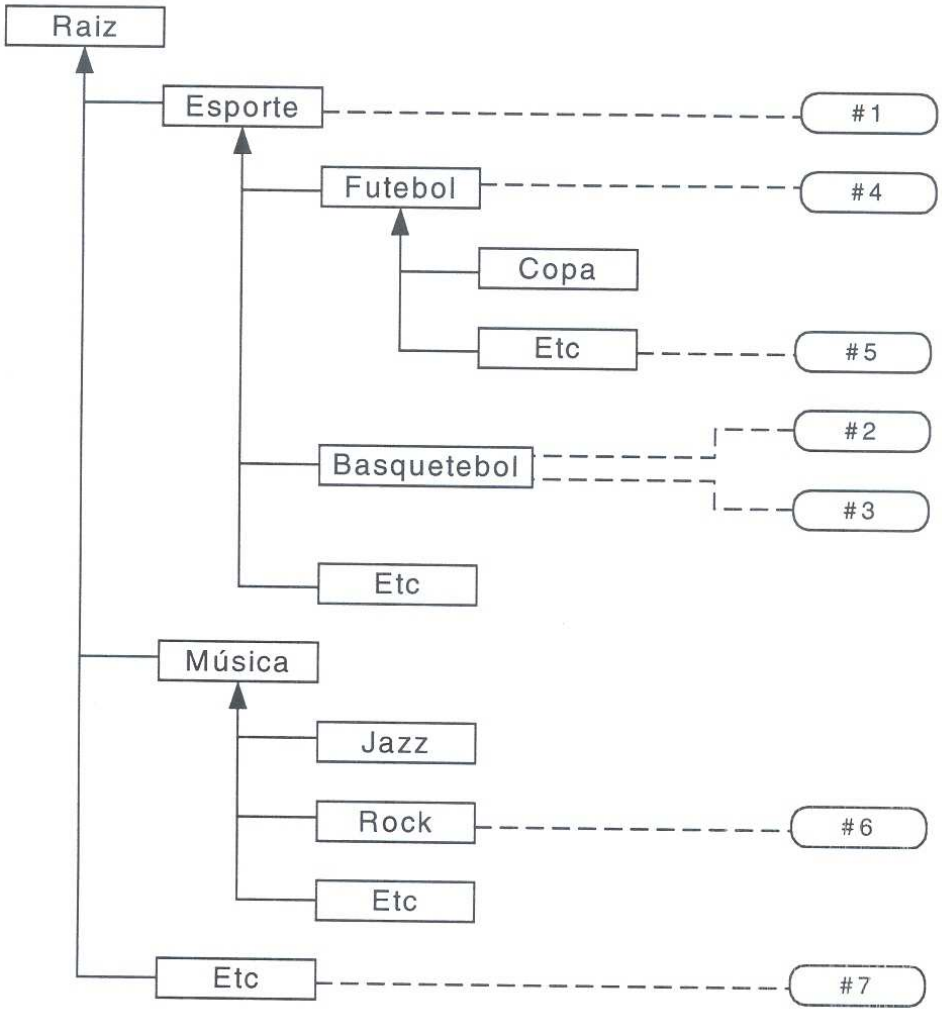


Figura 4.1 Exemplo de árvore semântica

4.2 Refinamento do algoritmo LSR

Os procedimentos invocados pelo procedimento *cria_espaço* (Figura 3.1, linha 10) são específicos da estrutura utilizada para classificação da informação. Esses procedimentos podem ser implementados da seguinte forma:

- O procedimento *extrai_semântica* (Figura 3.1, linha 4) supõe que cada objeto de informação carrega, além da própria informação, uma seqüência de assuntos e sub-assuntos, de acordo com a hierarquia; esta seqüência identifica a própria semântica do objeto. Tal suposição não é irrealista, visto que o padrão para a definição de objetos de informação na *Web*, o *XML* [*Light*; 1999], prevê esse tipo de informação (meta-informação) sobre o objeto.

- O procedimento *defina_local* (Figura 3.1, linha 5) recebe a semântica do objeto, como parâmetro, no formato de seqüência de assuntos, conforme discutido para o procedimento *extrai_semântica*. Com isso, o trabalho do procedimento *defina_local,* em uma árvore, fica trivial: basta seguir a seqüência até o seu final, pois para cada assunto há um nó correspondente.
- Os procedimentos *marque_assuntos_ancestrais* (Figura 3.1, linha 13) e *desmarque_assuntos_ancestrais* (linha 24) para um local da árvore, isto é, para um certo nó correspondente a um assunto, também têm implementação trivial em uma árvore: basta marcar ou desmarcar, respectivamente, o próprio nó e todos os seus ancestrais, recursivamente, até chegar à raiz da árvore.
- O procedimento *determine_objetos_mais_distantes* é mais complexo, e um pseudo-código está descrito na Figura 4.2. O procedimento supõe que para cada nó haja uma informação denominada *altura real* por indicar quantos níveis de descendentes existem, de forma que o descendente mais longínquo tenha pelo menos um objeto de informação vinculado, sendo que o próprio nó é considerado em sua *altura real*.

```
ConjuntoDeObjetos determina_objetos_mais_distantes (in Cache C, in Local l)
-- Determina o conjunto de objetos no cache C de menor relação semântica
-- (maior distância semântica) com relação ao local l do cache, que indica um assunto.
INÍCIO
    alvo := assunto raiz;
    Candidatos := conjunto de assuntos descendentes do alvo com altura real > 0;
    -- alvo possui algum descendente (direto ou indireto) que tenha associado
    -- pelo menos um objeto de informação ? Ou, equivalentemente, altura real do alvo > 1 ?
    ENQUANTO (Candidatos não vazio)
    FAÇA
    INÍCIO
        eleito: = selecione o elemento de Candidatos com a maior altura real;
        ENQUANTO(eleito estiver marcado como "ancestral"
            E existe um elemento de Candidatos que não seja o eleito)
        FAÇA
            --remove o eleito de Candidatos
            remove (Candidatos, eleito);
            --elege um novo elemento de Candidatos
            eleito := selecione o elemento de Candidatos com
                a maior altura real;
        alvo := eleito;
        Candidatos := conjunto de assuntos descendentes do alvo com altura real > 0;
    FIM
retorne o conjunto de objetos associado ao alvo;
FIM
```

Figura 4.2 Procedimento determina_objetos_mais_distantes

4.3 Exemplo de aplicação do LSR

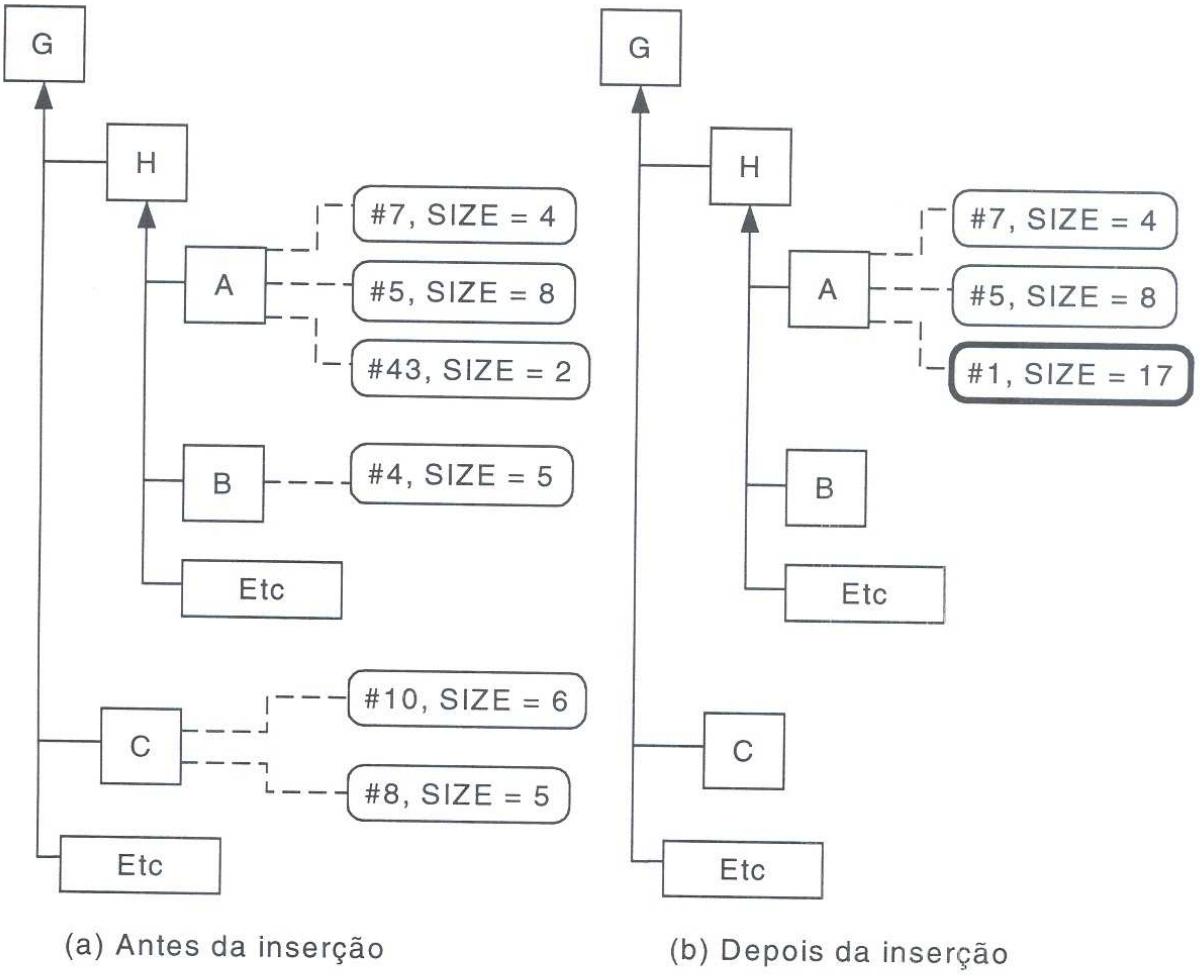


Figura 4.3 Árvore semântica antes e depois da inserção de um objeto

A Figura 4.3 mostra uma árvore semântica antes (a) e depois (b) da inserção de um novo objeto de informação. Este novo objeto possui identidade #1, semântica G.H.A e tamanho 17Kb. O tamanho máximo do cache neste exemplo é 30Kb, sendo que, antes da inserção, tem um espaço ocupado de 28Kb. Logo, será necessário remover um ou mais objetos para que o novo objeto seja inserido. Mais precisamente, será necessário remover objetos de tal forma que seja criado um espaço de pelo menos 15Kb, calculado por 28+17-30, isto é, *espaço ocupado + tamanho do novo objeto – tamanho do* cache. Aplicando-se o LSR, serão removidos do cache, pela ordem, os objetos #10, #8, #4 e #43, liberando um espaço total de 16Kb. Na verdade, a ordem de remoção entre os objetos #10 e #8 vai depender do algoritmo S empregado, pois este tem a mesma distância semântica do local de inserção (G.H.A).

4.4 Cenários de aplicação do algoritmo LSR

A partir desse cenário será apresentamos os passos para a aplicação do algoritmo LSR.

Cenário 1: Nesse cenário, o espaço do cache está ocupado por objetos, sendo que o tamanho máximo do cache, em *bytes*, é de 23000. O próximo passo é então descobrir o objeto candidato a ser removido do cache, quando da chegada do novo objeto, sendo que o *espaço livre* no cache é de 88 *bytes* (Figura 4.4). O objeto de informação a ser inserido no cache é o *#18,* de acordo com a semântica *Futebol, e* o tamanho do novo objeto em *bytes* é de *2078.*

- **Passo 1** - Verificar se o objeto *n* é maior que o tamanho total do cache, se o tamanho do objeto *n* for maior, o procedimento deverá ser anulado.
- **Passo 2** - Neste passo, o **LSR** verifica se o tamanho do novo objeto é maior que o *espaço livre* em cache e se existe um objeto distante semanticamente cujo tamanho seja suficiente para completar o espaço requerido. Portanto, o objeto candidato a ser removido deverá ser o de menor tamanho.
- **Passo3** - O tamanho do novo objeto é maior que o espaço livre e existe um objeto distante semanticamente, cujo tamanho é suficiente para completar o espaço requerido. O procedimento do LSR marca os assuntos_ancestrais para um local da árvore, isto é, para um certo nó correspondente que receberá a semântica do novo objeto ***ESPORTE.FUTEBOL***. A trilha de prioridade (caminho da inserção) marca o conjunto de nós correspondentes para garantir a permanência dos objetos com a semântica próxima do novo objeto.
- **Passo 4** - Quando ocorrer a marcação da prioridade, o algoritmo necessitará de um conjunto de assuntos descendentes do alvo ***Root***, com altura real > 0. Os objetos, eleitos como alvos descendentes (direto ou indireto) que tenham associado pelo menos um objeto de informação no qual a altura real do alvo seja maior que 1, (Figura 4.4).
- **Passo 5** - O algoritmo determina o elemento candidato com a maior altura real. Caso o elemento marcado como "*ancestral*" seja o escolhido e exista um elemento de candidatos que não seja o eleito, o próximo elemento com a segunda maior altura será o candidato. Neste cenário, o alvo *MÚSICA* será o escolhido, sendo que o primeiro elemento com a maior altura é *ESPORTE,* e ele não pode ser o elemento candidato em função da marcação de prioridade.

- **Passo 6 -** Definido o alvo *MÚSICA* como candidato, deve-se eleger o próximo candidato com a maior altura dentre os descendentes do alvo. Os descendentes do alvo com altura real > 0 são Etc, *JAZZ* e *ROCK*. O alvo Etc, presente em todas as sub-árvores, abriga assuntos não previstos na correspondente sub-árvore. Aplicando o algoritmo *S,* o objeto candidato escolhido para ser removido do cache é o objeto #13 do alvo Etc, pois, quando construída a árvore, fora eleito para ser o primeiro da lista quando as alturas reais dos descendentes forem iguais e trouxe necessidade de remover os objetos. O objeto ao ser removido criou espaço suficiente para que o novo objeto pudesse ser inserido no cache sem precisar remover outros objetos, o cenário final do cache após a inserção do novo objeto passou a possuir um espaço livre de 58 *bytes* (Figura 4.4).

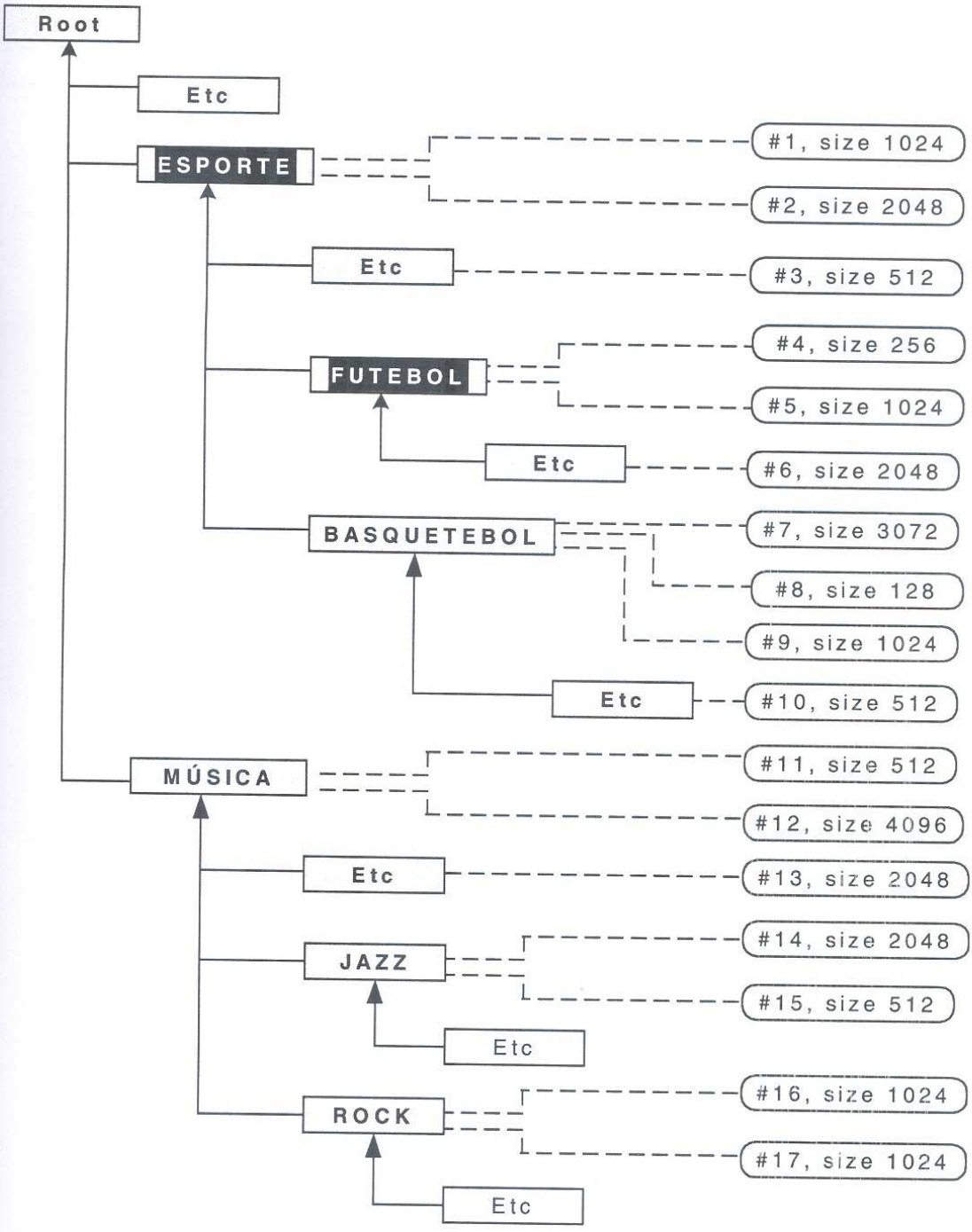


Figura 4.4 Árvore semântica em estado inicial

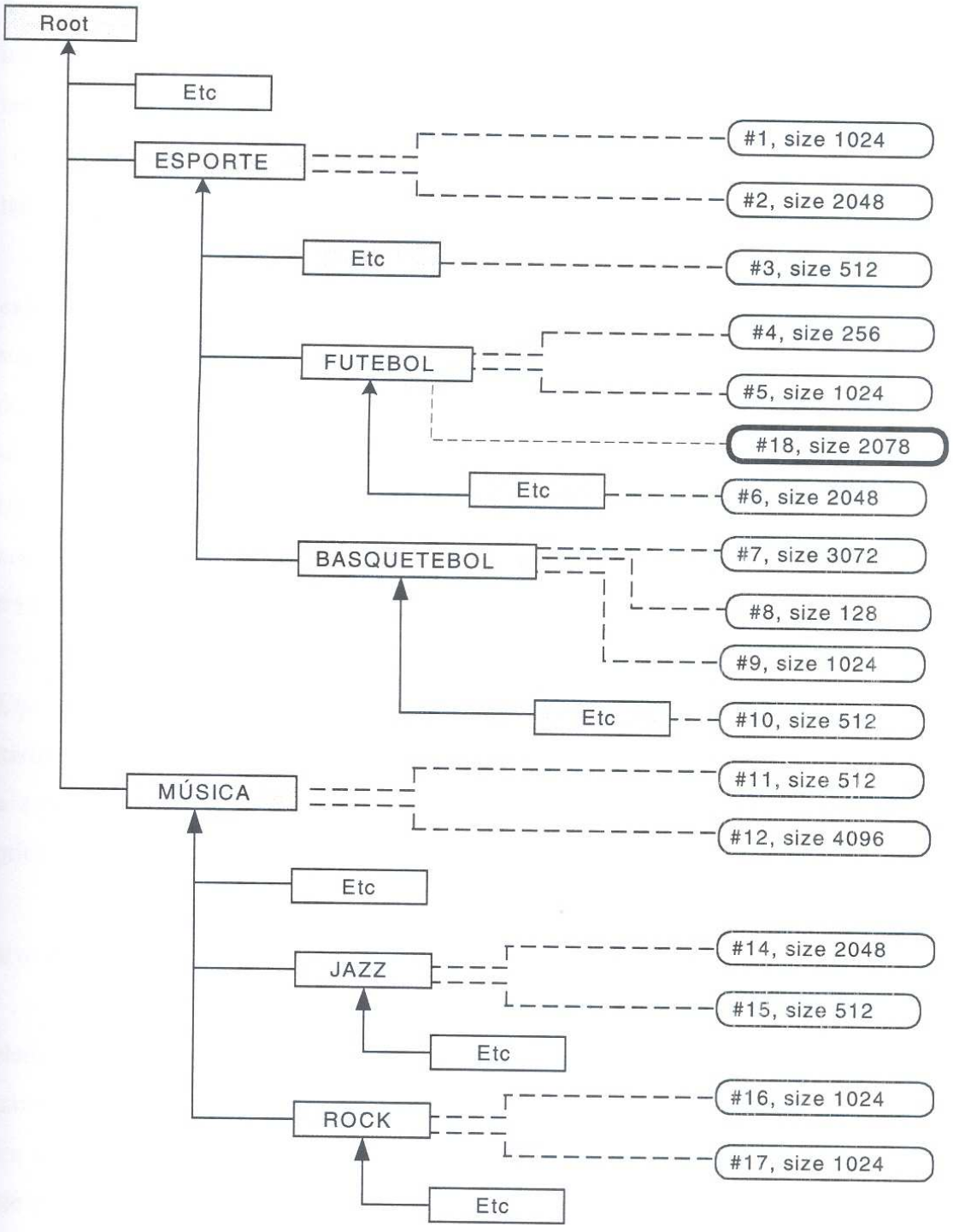


Figura 4.5 Árvore semântica em estado final

Cenário 2 - Descobre os objetos candidatos a serem removidos do cache, quando da chegada do novo objeto, sendo que o *espaço livre* no cache é de 58 *bytes* (Figura 4.5). O objeto de informação a ser inserido no cache é o *#19,* de acordo com a semântica *Basquetebol. O* tamanho do novo objeto em *bytes* é de *15360,*

- **Passo 1:** verificar se o objeto *n* é maior que o tamanho total do cache Se o tamanho do objeto *n* for maior, o procedimento deverá ser anulado.

- **Passo 2:** o **LSR** verificar o tamanho do novo objeto, isto é se ele é maior que o *espaço livre* em cache, e se existe um objeto distante semanticamente, cujo tamanho deva ser suficiente para completar o espaço requerido. Portanto o objeto, candidato a ser removido, deverá ser o de menor tamanho. Se o tamanho do novo objeto for maior que o espaço livre e se não existir um único objeto distante semanticamente, cujo tamanho seja suficiente para completar o espaço requerido, será então necessário remover objetos de mais de um nodo, mas nenhum objeto do nodo de inserção deverá ser removido em função do tamanho, não ser o suficiente para que isso ocorra.

- **Passo 3:** ao se inserir o objeto de informação *#19* com a semântica *Esporte.Basquetebol* e tamanho em bytes *15360*, o procedimento do LSR marca os *assuntos_ancestrais* para um local da árvore e para um nó correspondente, no qual vai receber a semântica do novo objeto. Neste passo, os objetos a serem marcados pelo vetor de prioridade são ***ESPORTE.BASQUETEBOL***.

- **Passo 4:** ocorre a marcação da prioridade e define os assuntos descendentes do alvo ***Root*** com altura real > 0.

- **Passo 5:** determinar os elementos candidatos a alvo de maior altura. Caso o elemento de maior altura esteja marcado como "*ancestral*", remover candidato eleito e eleger um novo elemento a candidato com a maior altura. O elemento candidato eleito como alvo, em nosso exemplo, é "*MÚSICA*", sendo que o primeiro alvo tinha sido *ESPORTE*, mas como ele estava marcado como ancestral, o passo seguinte foi o de escolher o segundo alvo com maior altura. Definido o candidato, o momento, agora, é o de definir, dentro do alvo "*MÚSICA*"; o conjunto de assuntos descendentes com altura > 0. Os elementos descendentes *JAZZ e ROCK* possuem altura =1. O procedimento determina que qualquer um dos dois descendentes, com a mesma altura, pode ser o eleito com alvo. O alvo eleito *JAZZ* recebe

aplicação do algoritmo *S* no conjunto de objetos vinculados à categoria, removendo do cache os objetos #14, em função do seu tamanho, e o objeto #15, por ser o outro objeto vinculado ao JAZZ, o que não foi suficiente para que o objeto #19 coubesse no cache.

- **Passo 6:** eleger o próximo alvo *ROCK,* pois ele faz parte do conjunto de assuntos descendentes do alvo "*MÚSICA*" e possui altura real > 0. O conjunto de objetos, vinculados à categoria a serem removidos do cache, aplicando o algoritmo *S,* são os objetos de informação #16 e #17. Neste exemplo, tanto faz retirar qualquer um dos dois objetos, pois eles possuem o mesmo tamanho, e o espaço criado, quando da sua remoção, não foi o suficiente para a entrada do objeto #19.

- **Passo 7:** retornando ao alvo *MÚSICA,* verificou-se que o conjunto de objetos associados ao alvo com altura real > 0 não existe. Então, o próximo elemento candidato é o próprio alvo *MÚSICA*, e o conjunto de objetos vinculados a categoria, a serem removidos do cache, aplicando o algoritmo *S* primeiramente, é o objeto #12 e depois o objeto #11, o que ainda não cria espaço suficiente para o objeto #19 possa entrar no cache.

- **Passo 8:** retornar e selecionar o elemento de candidatos com a maior altura real. Enquanto o espaço pendente não for o suficiente e não existir elemento de candidatos com grau > 0, deve-se desmarcar um dos assuntos_ancestrais, *ESPORTE*, e eleger o elemento candidato com a maior altura do conjunto descendente do alvo *FUTEBOL*. O próximo elemento com maior grau é o descendente Etc, o elemento alvo, possui grau > 0. Aplicando o algoritmo S, remove-se o objeto #6, o que ainda não cria espaço suficiente para o novo objeto.

- **Passo 9:** retornando ao alvo *ESPORTE,* eleger o próximo candidato com maior grau. O algoritmo LSR definiu o candidato eleito, neste caso, como o descendente de *ESPORTE,* o *node* Etc. Primeiramente, por possuir assunto menos semanticamente relacionado e, em segundo lugar por ter sido construído na árvore como anterior ao *node FUTEBOL*. Eleito o alvo, aplica-se o algoritmo S e retira o objeto #3 pois este ainda não criou espaço suficiente para o novo objeto

- **Passo 10:** retornando ao alvo *ESPORTE* e eleger o próximo candidato com maior grau de descendentes na categoria, neste passo, o eleito é *FUTEBOL*, portanto retiramos os objetos na seqüência #18, #5 e o #4 e ainda o espaço pendente não é suficiente para o novo objeto.

- **Passo 11:** retornando ao alvo *ESPORTE,* pode-se verificar que o conjunto de objetos associados ao alvo com altura real > 0, não existe. Então o próximo elemento candidato é o próprio alvo *ESPORTE*, e o conjunto de objetos vinculados a serem removidos do cache, aplicando o algoritmo *S,* primeiramente será o objeto #1, pelo fato de o espaço pendente ser o mínimo necessário para o novo objeto. A Figura 4.6 mostra o cenário 2, após a inserção do objeto #19

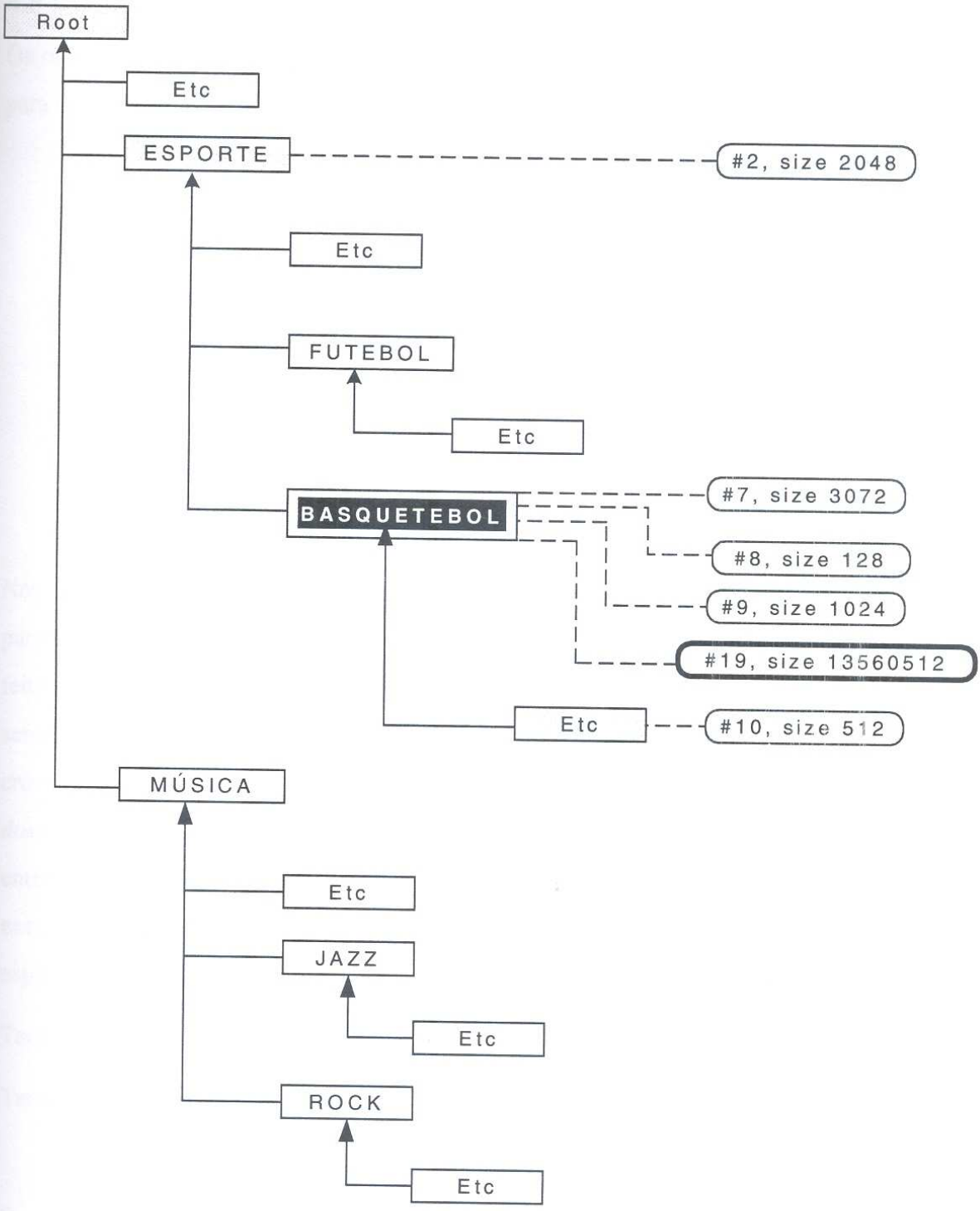


Figura 4.6 Árvore semântica com o objeto #19 inserido no *cache*

4.5 Demonstração da retirada do objeto do cache

Os objetos #14 e #15 do *cenário 2,* demonstrados na Figura 4.7, foram os objetos escolhidos para serem retirados da árvore semântica a fim de criar espaço para o novo objeto.

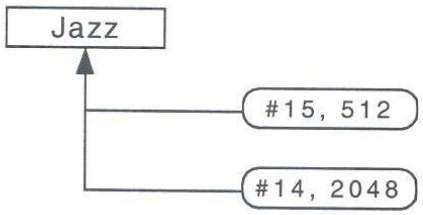


Figura 4.7: Nodo da árvore

Nesse exemplo, os dois objetos estão no mesmo nível semântico e o procedimento do LSR para a escolha do objeto a ser retirado em primeiro lugar vem de um verificador com o qual é feita a comparação entre o tamanho do novo objeto e o tamanho dos objetos eleitos para serem removidos do cache. Os objetos do cache são internamente posicionados em ordem crescente, pelo tamanho em byte, e a escolha é feita por *n.tamanho* do novo objeto – *tamanho dos objetos #15, e n.tamanho* do novo objeto – *tamanho dos objetos #14.* A mínima diferença entre o espaço pendente e o tamanho do novo objeto é escolhida para sair do cache. Neste exemplo, o objeto #15 foi eleito para sair do cache pois possuía a menor diferença entre o espaço pendente e o tamanho do novo objeto.

Tamanho objeto *n*	15360
Tamanho do objeto *#15*	-512
	14848
Tamanho objeto *n*	15360
Tamanho do objeto *#14*	-2048
	13312

CAPÍTULO 5

Experimento e validação

5.1 Esquema de simulação do LSR

O esquema para simular o LSR utiliza uma lista de caminhos, selecionado manualmente, da estrutura do *Yahoo*!. A lista de caminhos possui uma organização, na estrutura que faz com que as informações sejam baseadas em uma hierarquia (capítulo 4). A estrutura hierárquica, montada na simulação, classifica semanticamente os objetos. Os objetos, para serem classificados, devem vir de uma seqüência de acessos a objetos, obtido através do Método de Monte Carlo.

A Figura 5.1 mostra a lista *caminhos* e a *seqüência de acessos a objetos* , descritos parcialmente. O simulador do LSR permite a obtenção de gráficos para avaliação de ocupação de cache e a taxa de acerto (*hit ratio*). As seções subseqüentes explicam em detalhes cada um dos elementos da Figura 5.1

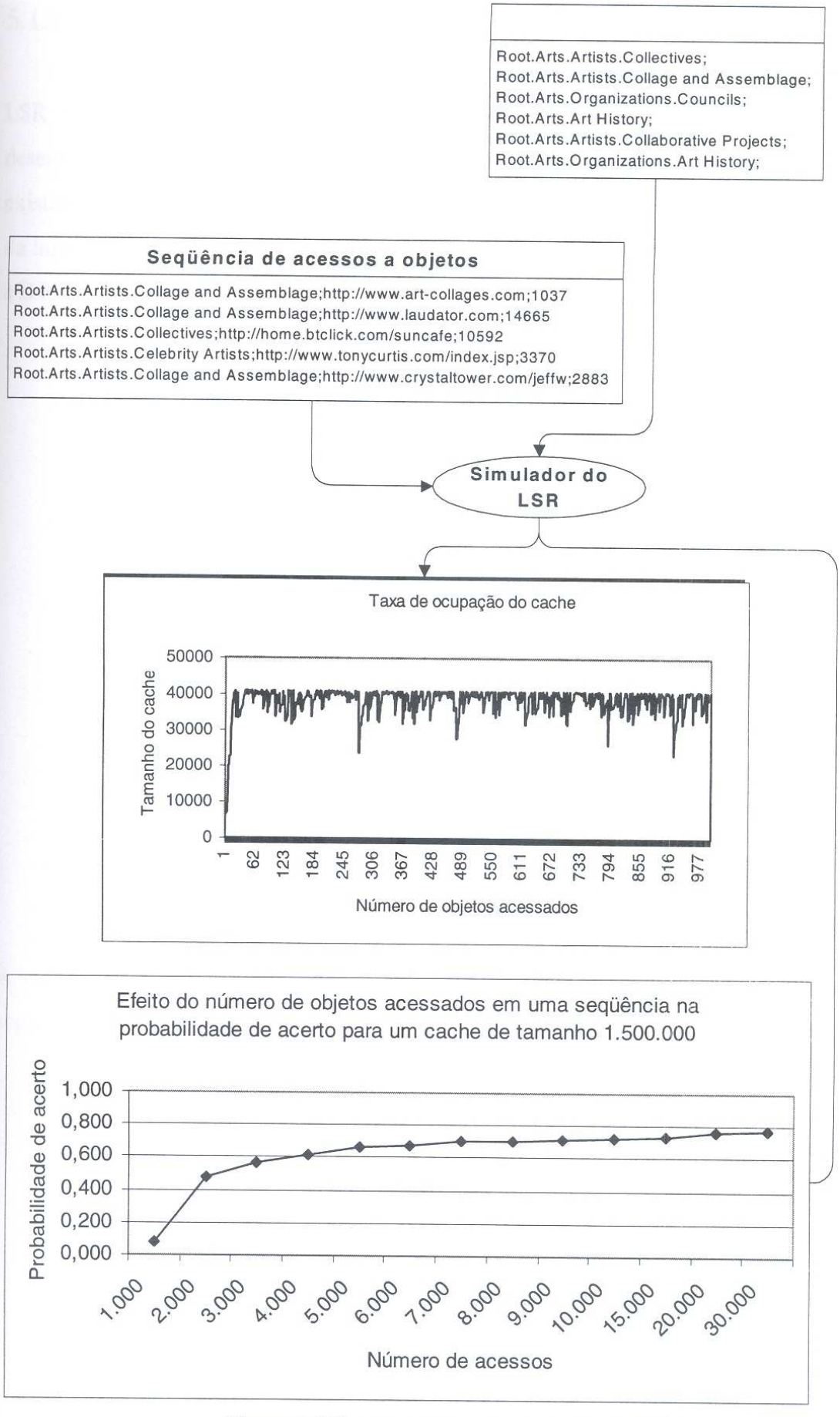


Figura 5.1 Esquema básico de simulação do LSR

5.1.1 Representação de categorias

Baseado no esquema de hierarquia descrito em seções anteriores, a implementação do LSR foi desenvolvida conforme o exemplo da Figura 5.2. A hierarquia das categorias foi desenvolvida através da raiz ***Root***, que é o *node* inicial para todos os assuntos das categorias existentes (caminho). As categorias sob o ***Root*, X e Y** foram inseridas conforme a semântica da informação. Para a categoria da semântica **X** foram criadas as categorias de informações **A** e **B** e sob a categoria **Y** foram criadas as categorias de assuntos **C** e **D**.

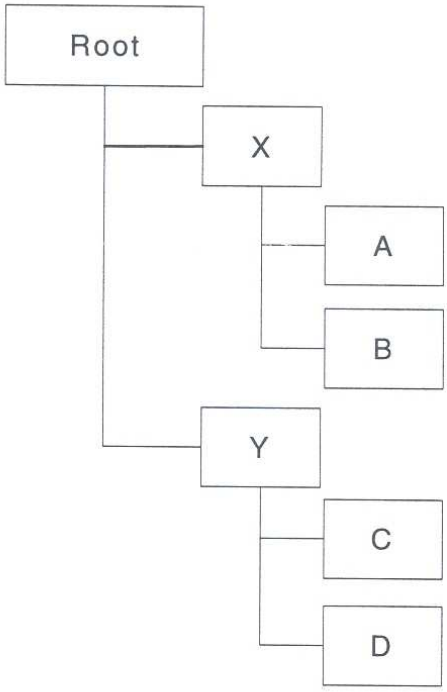


Figura 5.2. Hierarquia de categorias

A hierarquia, vista na Figura 5.2, é representada textualmente na relação de caminhos que segue.

```
Root.X;
Root.Y;
Root.X.A;
Root.X.B;
Root.Y.C;
Root.Y.D;
```

5.1.2 Seqüência de acessos a objetos

Os objetos inseridos na hierarquia da Figura 5.3, identificados por #7, #12, #15, #3, #10, estão posicionados nos seus respectivos *nodes* (categorias de assuntos), de acordo com a semântica de cada objeto de informação. Uma seqüência de acessos a objetos é obtida a partir dos objetos contidos em cada nó. Assim, cada item da seqüência deve descrever o objeto acessado completamente, isto é, o caminho de *nodes* onde o objeto esta inserido, a identificação do universo (normalmente a sua URL) e o tamanho do objeto normalmente, (em *bytes*), como mostra o exemplo que segue.

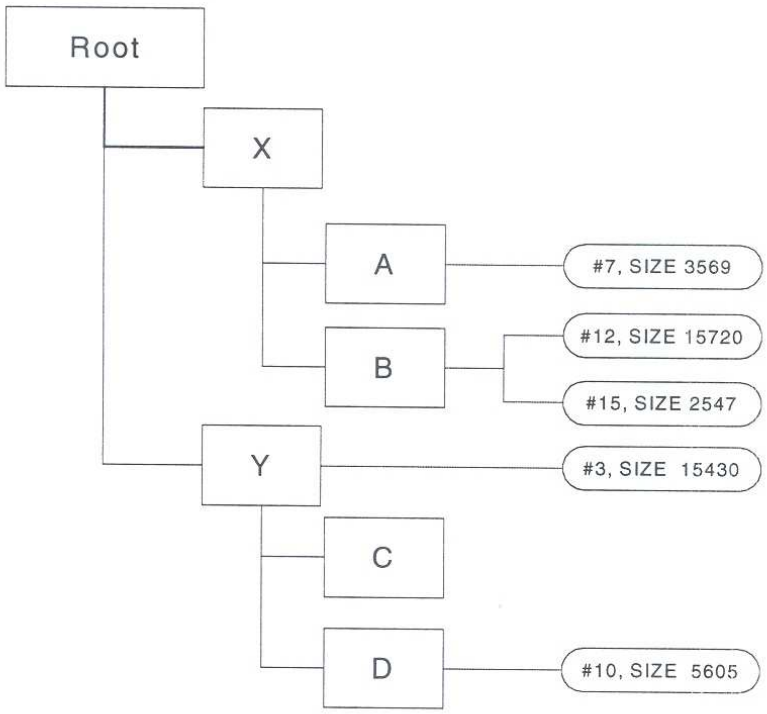


Figura 5.3: Hierarquia de categorias com objetos de informação e tamanho

Seqüência de acessos a objetos
Root.X.A;#7;3569
Root.X.B;#15;2547
Root.X.D;#10;560
Root.X.B;#12;15720
Root.X.A;#7;3569
Root.X.B;#15;2547
Root.X.D;#10;560
Root.X.B;#12;15720

5.1.3 Taxa de ocupação do cache

A taxa de ocupação do cache é demonstrada através do gráfico da Figura 5.4, que representa a simulação de 1.000 acessos a objetos, com o tamanho do cache fixado em 4.000.000 *bytes*. Podemos verificar que a taxa de ocupação do cache esteve,em sua maior parte, perto do limite máximo do cache, proporcionando uma taxa de ocupação estável ocupação de objetos em cache.

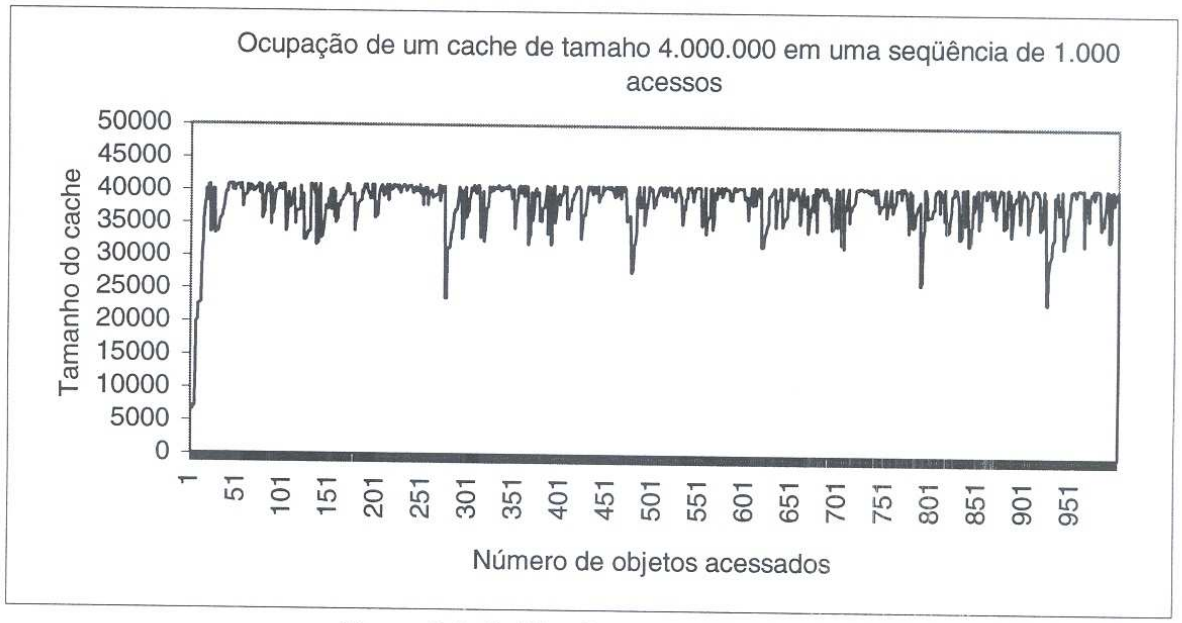
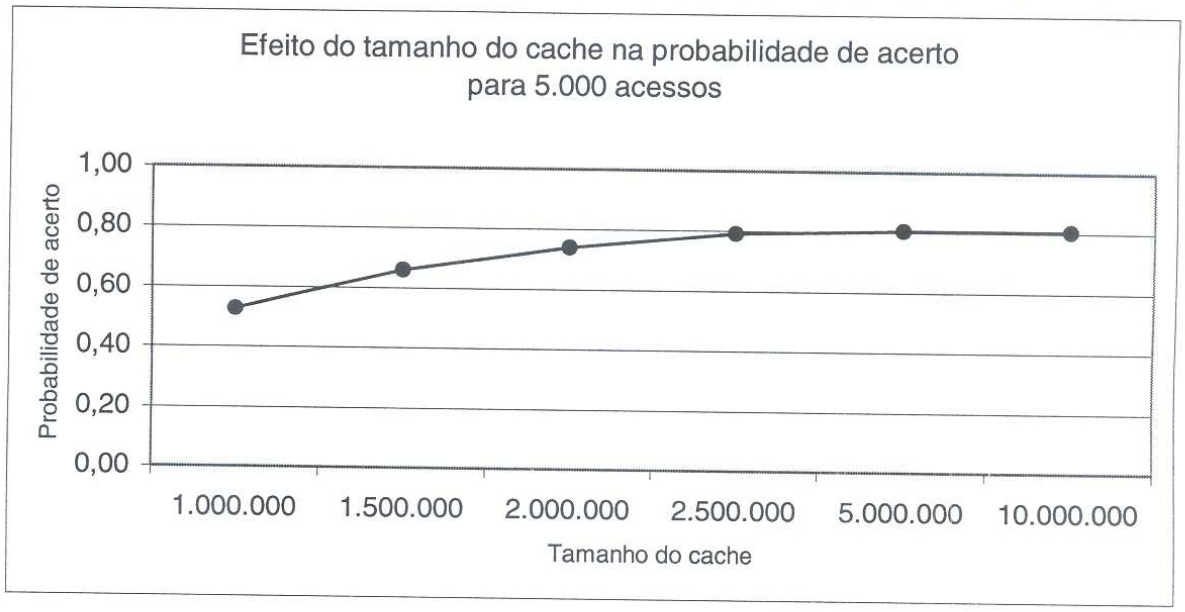


Figura 5.4: Gráfico de ocupação do cache

5.1.4 Probabilidade de acerto

A eficiência de um cache pode ser medida pela probabilidade de acerto, que é a ocorrência de um acesso a um objeto já presente no cache. Tal probabilidade pode ser obtida em função da variação do tamanho total do cache ou em função do número de acessos, como mostra a Figura 5.5.



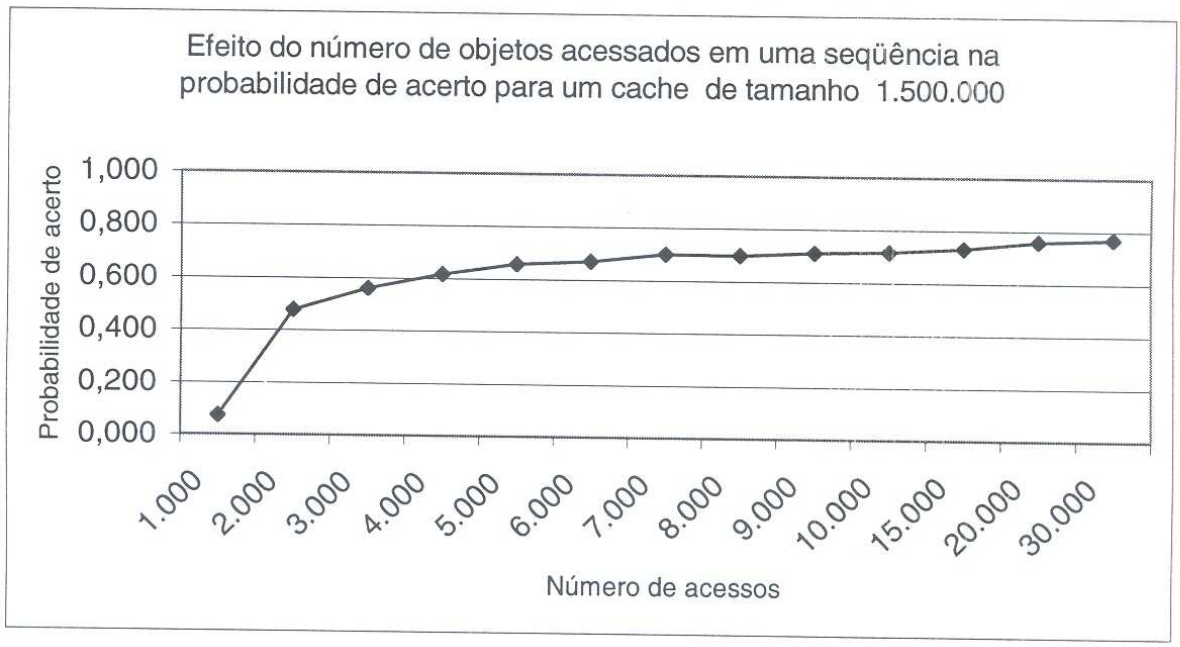


Figura 5.5: Gráfico de probabilidade de acerto

5.1.5 Simulador do LSR

O algoritmo LSR foi implementado para fins de validação e comparação, conforme a arquitetura apresentada na Figura 5.6. Um módulo denominado *Cliente Web* representa qualquer entidade da *Web* que utilize um cache, como um *browser*, um *proxy* ou até mesmo um servidor *Web*.

O módulo *Cliente Web* acessa uma seqüência de objetos de informação. Cada objeto de informação possui dois atributos: sua semântica e seu tamanho. Para essa seqüência não há qualquer preocupação de estrutura ou ordenação.

O módulo *Gerenciador do* Cache contém o cache onde são armazenados os objetos de informação e uma árvore de classificação semântica dos objetos. Todo o controle do cache, isto é, a implementação do algoritmo LSR, é feito por este módulo.

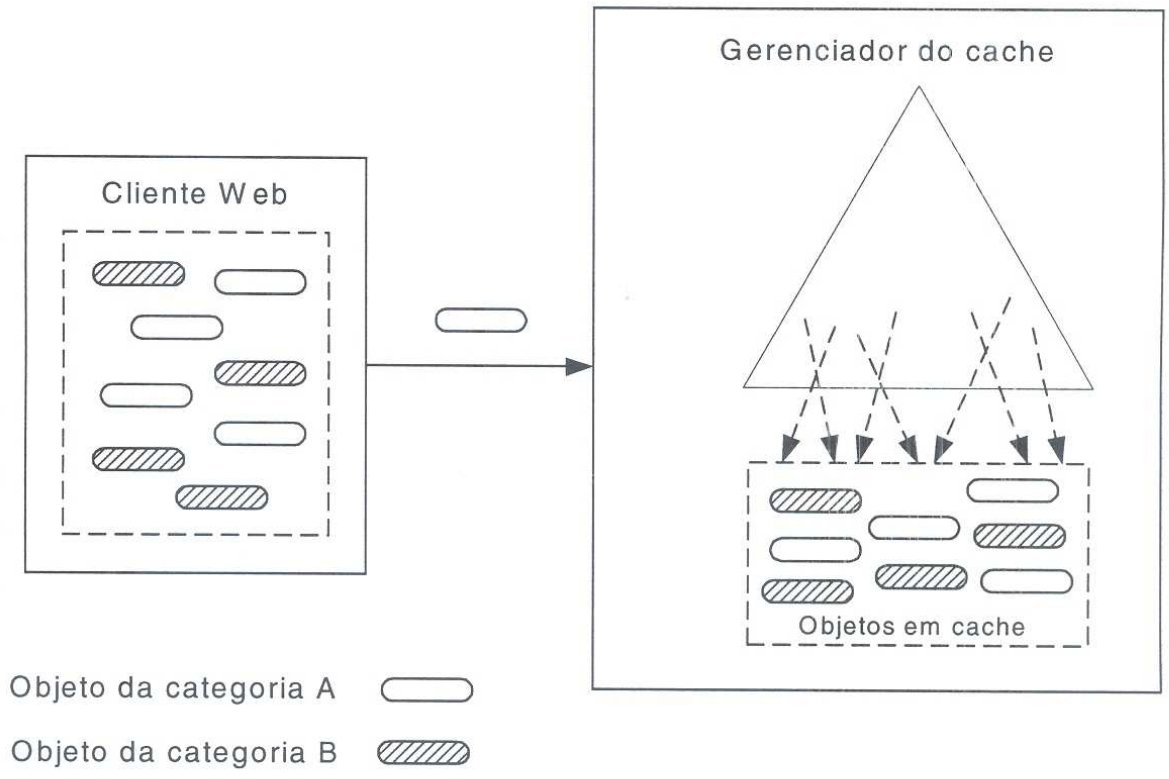


Figura 5.6. Arquitetura do simulador do LSR

5.2 Classificação semântica de objetos na Web

Os mecanismos de pesquisa on-line permitem que você encontre informações sobre assuntos específicos de forma rápida e eficaz. Hoje os usuários da *Web* começam suas pesquisas por *sites Web* específicos, que oferecem mecanismo de pesquisa. Usando estes mecanismos para pesquisa, os usuários podem localizar rapidamente *sites* que tratam de assuntos de seu interesse. Atualmente, dentre os mecanismos de pesquisa mais populares da Internet estão o *AltaVista, Excite, InfoSeek, Lycos, WebCrawlwr e Yahoo!* (ver Anexo II).

Conforme foi dito anteriormente, os mecanismos de pesquisa capacitam os usuários para pesquisar na *Web* documentos que contenham o assunto de seu interesse. A maior parte dos mecanismos de pesquisa lê os documentos, baseados na Internet, regularmente e armazena informações sobre cada documento no banco de dados local do mecanismo de pesquisa. Por exemplo, o mecanismo de pesquisa *Yahoo*!, em 1998, indexou cerca de mais de 30 milhões de documentos na *Web,* em mais de 20.000 categorias, residentes em 275.600 servidores, e 4 milhões de artigos de 14.000 *newsgroups* da *Usenet* [*Peter Dyson* 1998][2]. Os usuários da Internet acessam o *site* do *Yahoo*! mais de 18 milhões de vezes por dia.

Para encontrar informações disponíveis na internet, usando o *Yahoo*!, os usuários têm de digitar o assunto que procuram (ver Anexo III). O *Yahoo*!, por sua vez, consulta seu banco de dados e apresenta uma breve descrição de todos os sites da Internet, juntamente com os respectivos URLs, que mais se assemelham ao conteúdo solicitado. Por exemplo, ao pesquisar o assunto "***Photography***", o *Yahoo*! trará milhares de documentos que mais se assemelham ao conteúdo solicitado, classificado pela probabilidade do site conter a informação procurada. Neste caso, o primeiro site listado corresponde ao site Web, http://www.nd.edu/~crosenbe/jobs.html. O *Yahoo*! traz, ainda, consigo uma série de categorias de assuntos relacionados com ***Photography***, entre eles ***Daguerreotypes, Historic Collections***, etc.

Os assuntos são associados a um ou mais assuntos mais genéricos e, por isso, podem aparecer uma ou mais vezes na estrutura do *Yahoo*!. É o caso do assunto "***Photography***", como mostra a *Figura 5.8.* Esse assunto encontra-se sob o assunto "***Art History***" e sob o assunto "***Artists.Group Exihibits***", *e sob* "***Artists,*** bem como ***Jounals*** sob ***Humanities*** e sob

---

[2] Dominando o Microsoft Internet Information Server Internet / Intranet "A Bíblia"

Humanities.Literature. Na versão atual do LSR, essa repetição não pode ocorrer em uma hierarquia de assuntos (categorias). *A Figura 5.9* mostra a hierarquia de categorias do LSR correspondente à hierarquia do *Yahoo*!, juntamente com os *nodes Etc* para assuntos não relacionados semanticamente com a estrutura hierárquica.

Os assuntos associados nas categorias do LSR não devem aparecer mais de uma vez, em função de política adotada pelo LSR não tratar deste tipo de informação. As repetições do mesmo assunto em categorias diferentes não estão sendo tratadas, nesse momento, por esta versão do LSR, ficando o mesmo assim, para eventuais trabalhos futuros. Exemplo do *Yahoo*!: repetição da categoria *Photography* Figura 5.7.

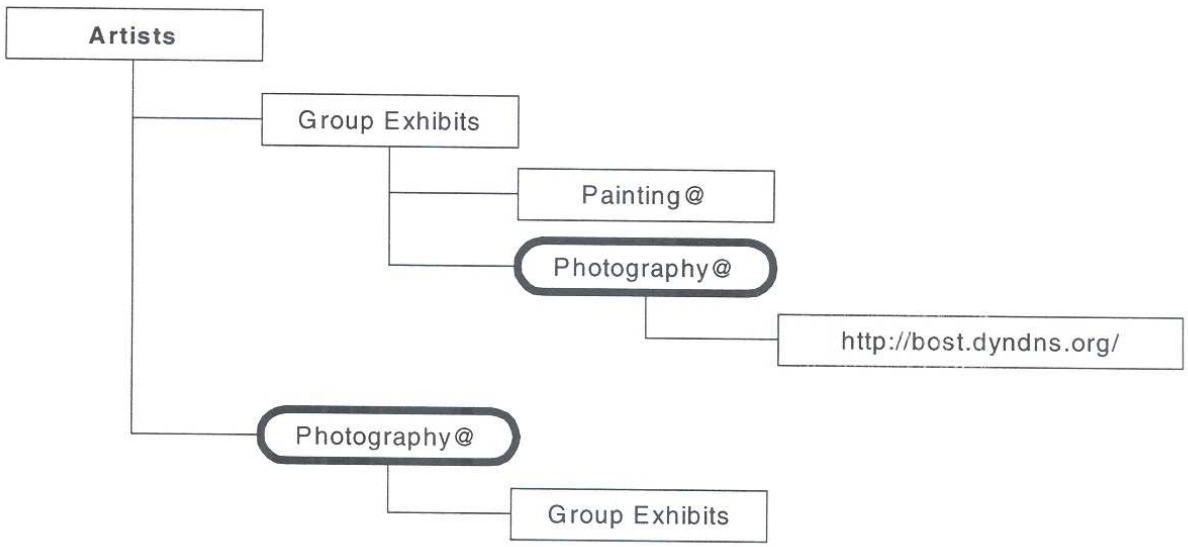


Figura 5.7 Hierarquia parcial da categoria *Artists* do *Yahoo*!

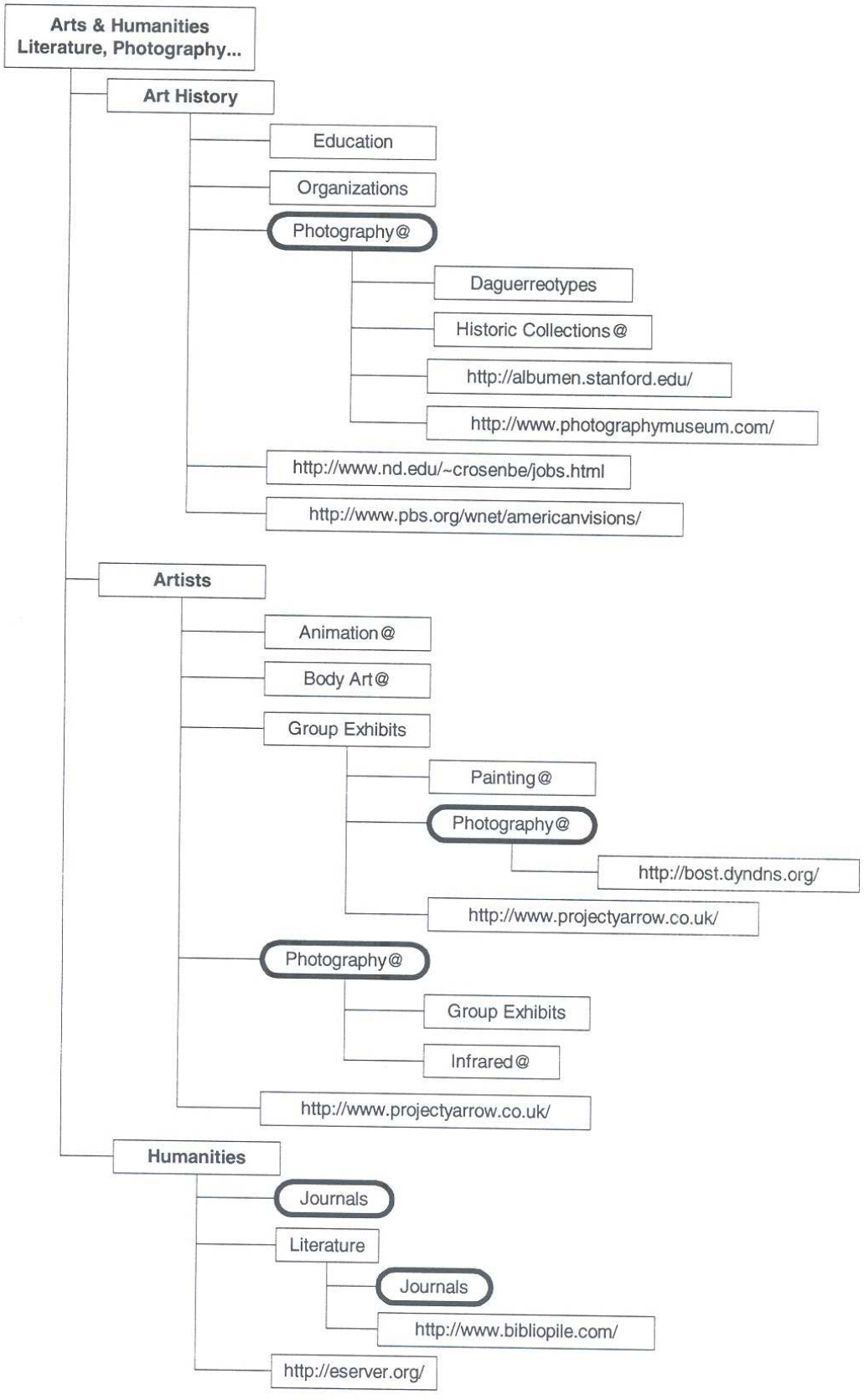


Figura 5.8: Hierarquia parcial da categoria *Arts & Humanities* do *Yahoo!*

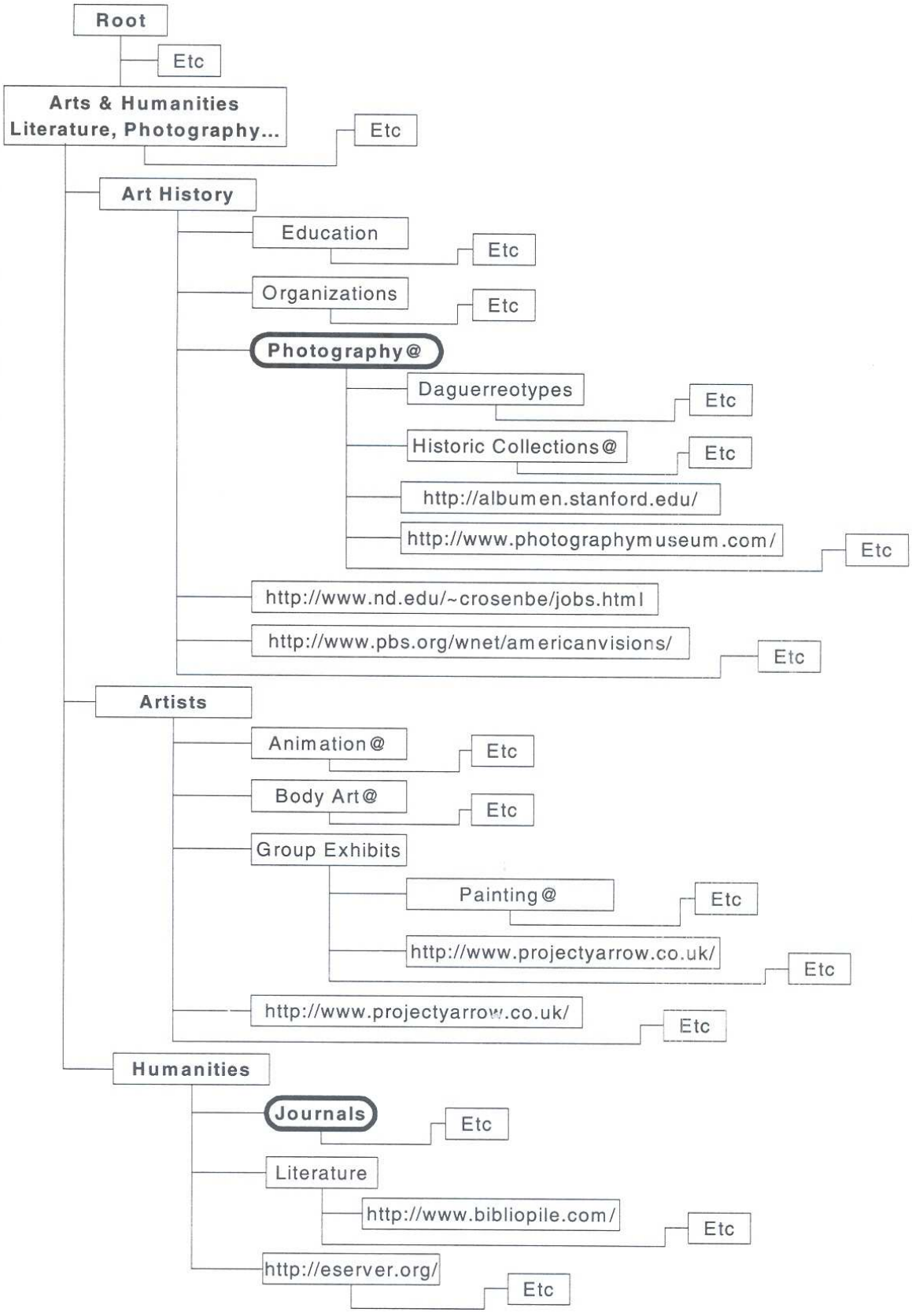


Figura 5.9: Hierarquia do LSR correspondente à hierarquia do *Yahoo*! da Figura 5.8

5.3 Preparação de dados para a simulação de cache

O esquema mostrado na Figura 5.10 contém uma seqüência de passos para preparação de dados para simulação de cache.

Durante um determinado período, foram utilizados os acessos a objetos registrados pelo *Proxy Squid* da *Pontifícia Universidade Católica do Paraná,* cerca de 2.000.000 de registros. As informações, vindas do *proxy* foram separados por URLs, que por sua vez foram separadas pela quantidade de acessos, que indica a freqüência com que os usuários acessaram os objetos no período. A partir das URLs com acessos, foi possível produzir uma distribuição da quantidade de acessos pela quantidade de objetos, demonstrada na Figura 5.11. As seções subseqüentes detalham o processo de preparação dos dados

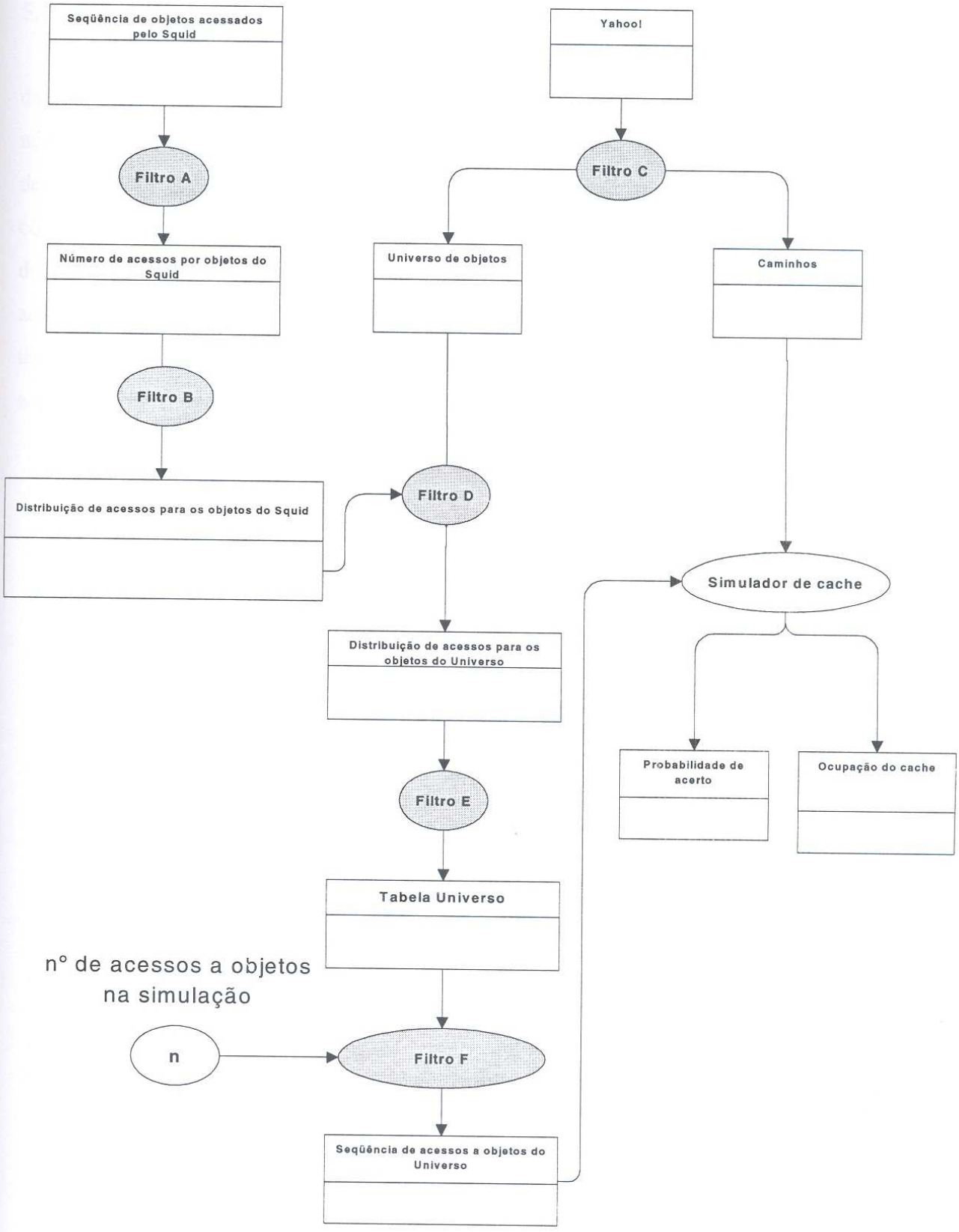


Figura 5.10: Esquema de preparação de dados para a simulação de cache

5.3.1 Obtenção da distribuição de acessos

A distribuição da quantidade de acessos pela quantidade de objetos, vindos do *proxy* da PUCPR, foram essenciais para que se pudesse fazer a obtenção e a distribuição do maior número de objetos acessados pelos usuários. A Figura 5.11 mostra parcialmente a seqüência de objetos acessados pelo *Squid* que, passando pelo Filtro[3] A, transforma-se numa seqüência com o número de acessos por objetos. Esta nova seqüência é submetida ao Filtro B que gera a distribuição de acesso para os objetos, definindo categorias de acesso e o número de objetos acessados em cada categoria. Nota-se, pela distribuição representada no gráfico, que há uma tendência de muitos objetos serem acessados poucas vezes e de poucos objetos serem acessados muitas vezes.

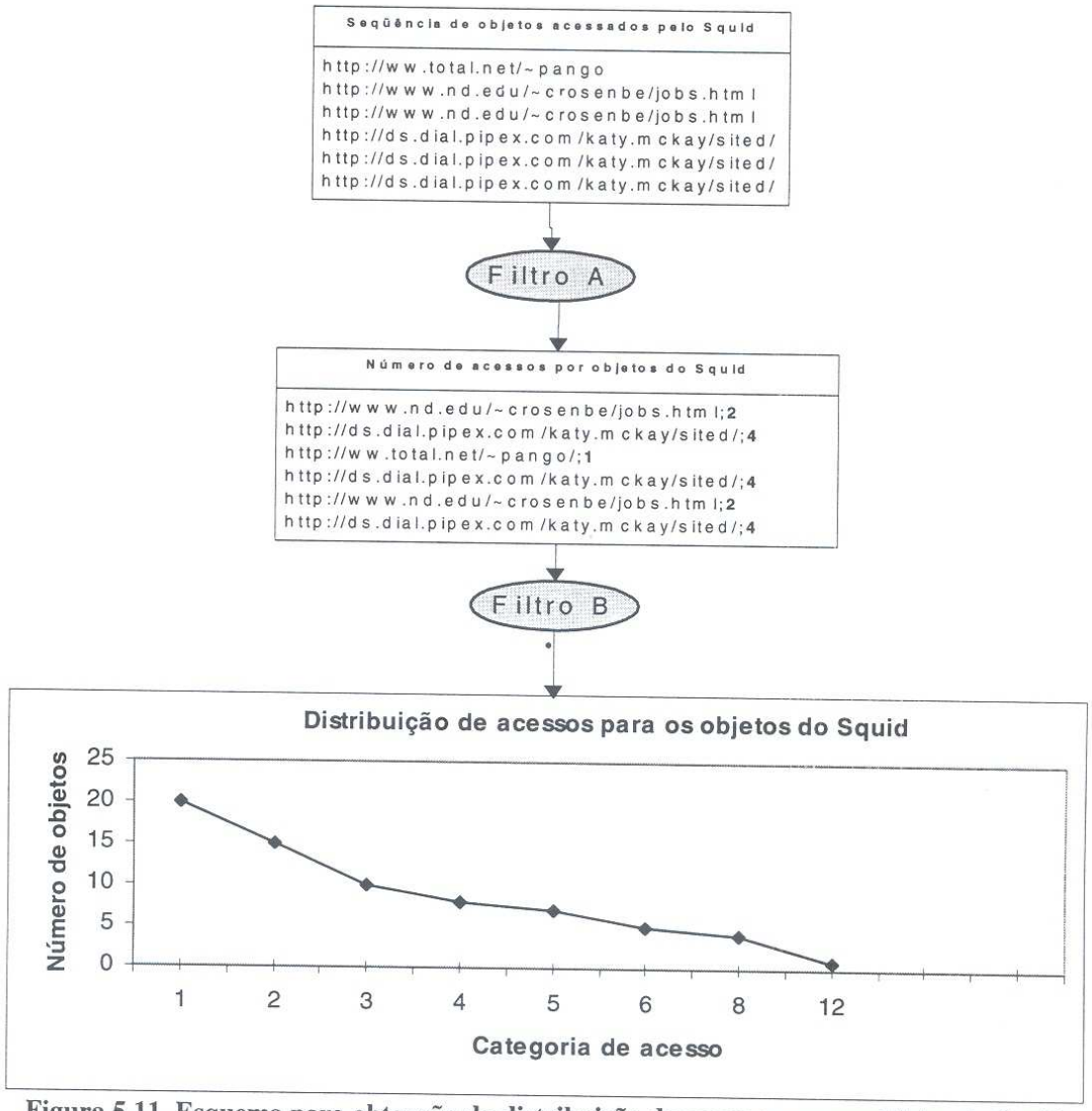


Figura 5.11. Esquema para obtenção da distribuição de acessos para os objetos do Squid

[3] Um Filtro é um programa que faz um arranjo em um conjunto de dados, segundo alguns critérios

O Anexo IV.1 mostra a seqüência completa de objetos acessados pelo *Squid* num certo período. O Exemplo 2 mostra a correspondente seqüência com números de acessos, obtida com a aplicação do Filtro A. O gráfico de distribuição de acessos em categorias correspondente, isto é, obtido pela aplicação do Filtro B, é o próprio gráfico representado na Figura 5.11.

5.3.2 Obtenção do Universo de objetos

A Figura 5.12 ilustra o esquema utilizado para obtenção de um conjunto de objetos com suas respectivas semânticas, que permitisse a simulação de cache. Tal conjunto foi denominado **Universo de Objetos**, enquanto que o conjunto de todos caminhos (os quais definem semântica de objetos) correspondentes a esses objetos foi simplesmente denominado de **Caminhos.** Cada elemento do Universo de Objetos é representado por três itens: (i) o caminho do objeto, (ii) a URL do objeto e (iii) o tamanho do objeto em bytes. Cada elemento de Caminhos é representado somente por um caminho de algum objeto contido no Universo de Objetos. Como visto na seção 5.4, uma base de objetos adequada para nosso experimento é fornecida pelo *site Yahoo*!, pois os objetos encontram-se devidamente classificados. O Filtro C foi implementado como um processo manual isto é, através da interação com o *site* do *Yahoo*!, são extraídos os conjuntos Universo de objetos e Caminhos. O Anexo IV.3 e o Anexo IV.4 mostram respectivamente um Universo de Objetos obtido a partir do *Yahoo*! e o correspondente conjunto Caminhos.

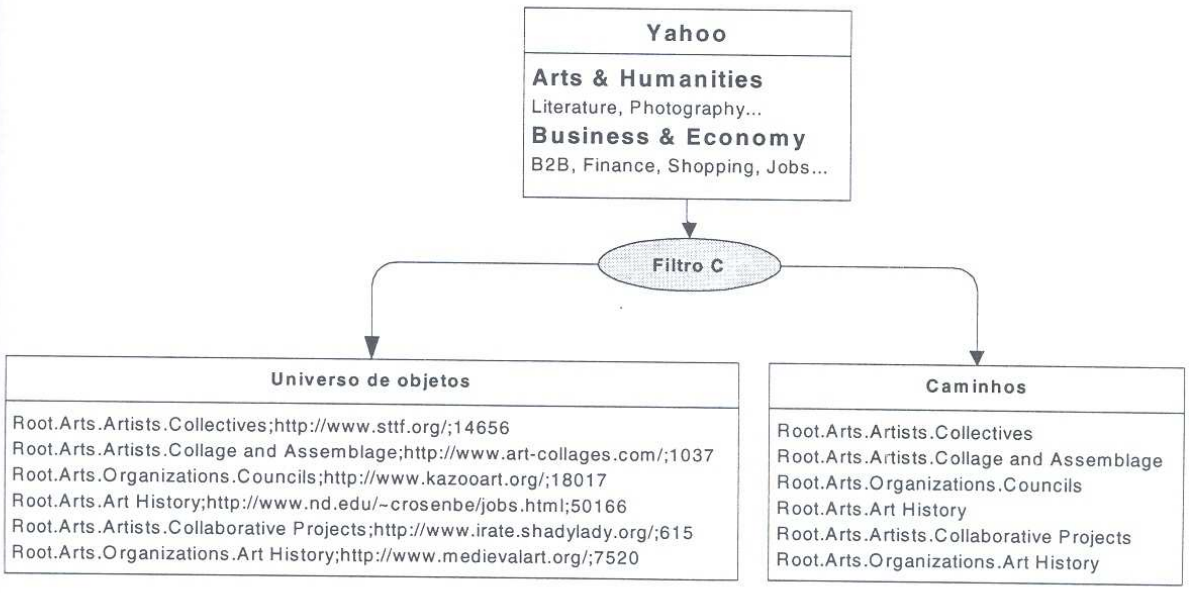


Figura 5.12: Esquema de obtenção do Universo de objetos

5.3.3 Obtenção da distribuição de acessos para os objetos do Universo

Conforme visto na Seção 5.3.1, existe uma distribuição em categorias dos objetos acessados num certo período de acordo com o número de acessos sofridos por cada um. Essa distribuição constatada na prática deve, portanto, ser reproduzida em um Universo de Objetos selecionado pela forma descrita na Seção 5.3.2. Dessa forma, cada entrada do objeto de Universo deve ser acrescida com a categoria de acessos correspondentes.

A Figura 5.13 ilustra o esquema para obtenção da distribuição de acessos para os objetos do Universo: o Filtro **D** reproduz a distribuição de acessos para os objetos do *Squid* em um certo período no Universo de Objetos, obtendo a categoria de acesso para cada objeto do Universo de acesso. O Anexo IV.5 mostra a distribuição de acessos para os objetos do Universo de acordo com a distribuição de acessos para os objetos do *Squid* da Figura 5.11.

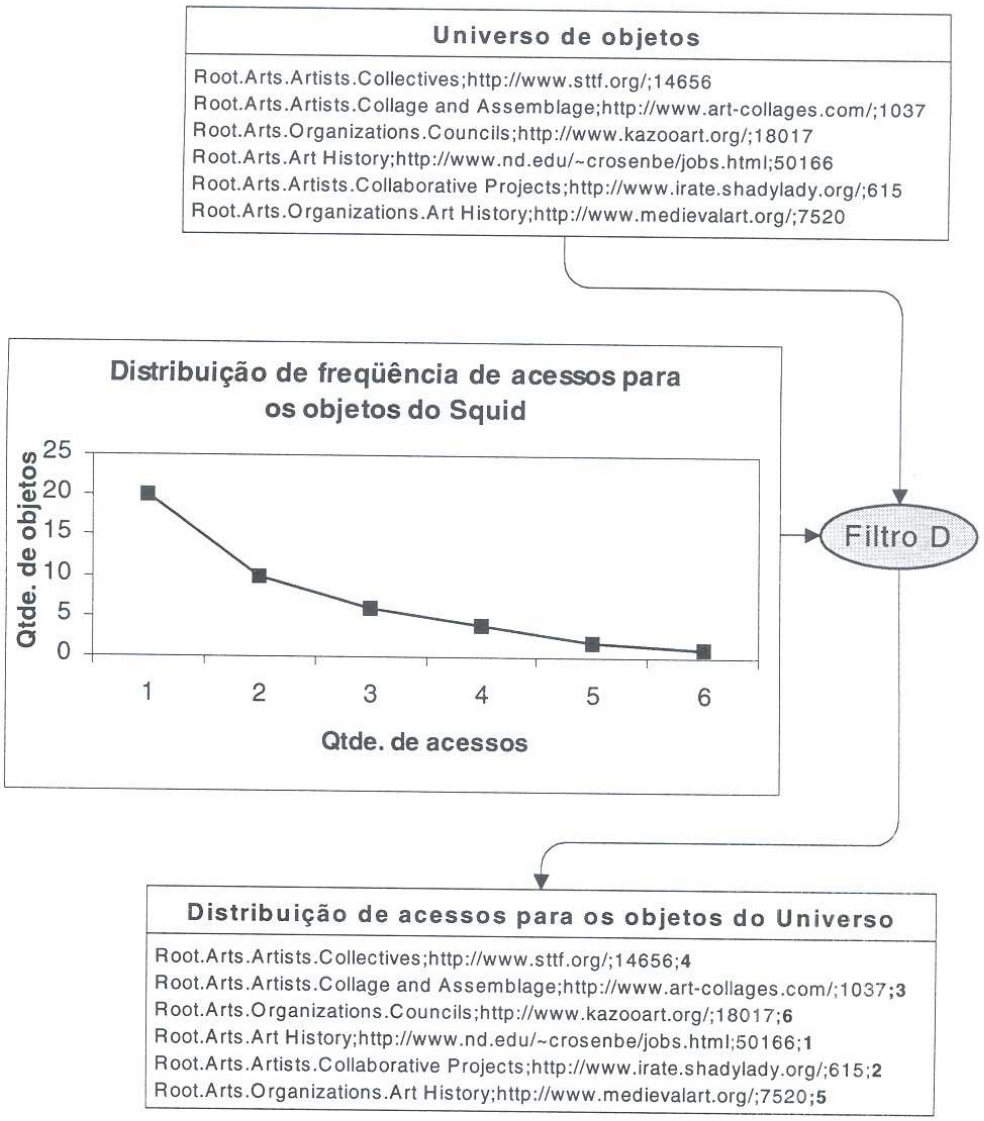


Figura 5.13: Esquema para obtenção da distribuição de acessos para os objetos do Universo

5.3.4 Obtenção da Tabela Universo

O objetivo do processo de preparação de dados para simulação de cache é obter uma seqüência de acessos a objetos do Universo selecionado. Essa seqüência deve, ao mesmo tempo, ter um fator de aleatoriedade, pois representa uma seqüência de acessos realizados por um usuários qualquer, e ainda obedecer à mesma distribuição de acessos constatada na prática, conforme visto na Seção 5.3.3. Para atender a esses requisitos será aplicado o Método de Monte Carlo, que será discutido na Seção 5.3.5. A aplicação desse método requer que os objetos do Universo estejam arranjados em categorias e que cada objeto tenha uma identificação única dentro da sua categoria. O Filtro E mostrado na Figura 5.14, faz tal arranjo e identificação, criando o que denomina-se **Tabela Universo**, a partir da distribuição de acessos para o objeto do Universo. O Exemplo 6 mostra a Tabela Universo correspondente à distribuição de acessos do Exemplo 5, obtida com aplicação do Filtro E.

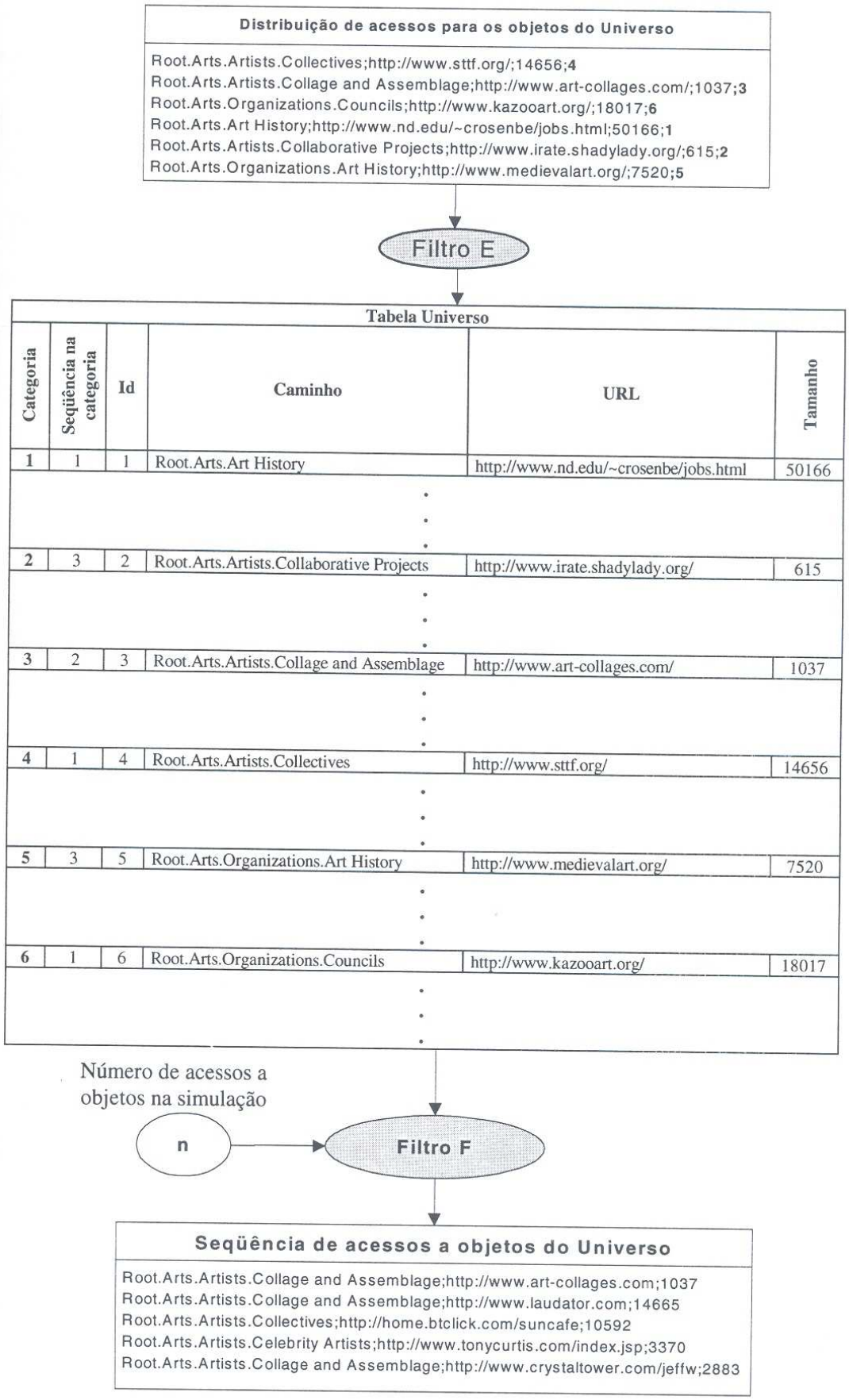


Figura 5.14: Esquema dos objetos com acessos

Exemplo 6: Tabela Universo correspondente à distribuição de acessos do Exemplo 5

Tabela Universo					
Categoria	Seqüência na categoria	Id	Caminho	URL	Tamanho
1	1	1	Root.Arts.Art History	http://www.nd.edu/~crosenbe/jobs.html	50166
	2	2	Root.Arts.Art History	http://www.pbs.org/wnet/americanvisions	6421
	3	3	Root.Arts.Art History	http://users.hol.gr/~dilos/anistor/cover.htm	1213
	4	4	Root.Arts.Art History.Architectural History	http://www.indiana.edu/~kglowack/athens	2338
	5	5	Root.Arts.Art History.Architectural History	http://library.thinkquest.org/10098	5744
	6	6	Root.Arts.Artists.Collage and Assemblage	http://www.flyingjeannie.com	438
	7	7	Root.Arts.Art History.Photography	http://www.photographymuseum.com	27869
	8	8	Root.Arts.Art History.Photography	http://www.pbs.org/ktca/americanphotography	927
	9	9	Root.Arts.Art History.Photography	http://www.primenet.com/~sos/photopage.html	3454
	10	10	Root.Arts.Art History.Thematic	http://www.celtic-twilight.com/artists	2192
	11	11	Root.Arts.Art History.Thematic	http://members.aol.com/alanta98/cardart.html	11355
	12	12	Root.Arts.Art History.Thematic	http://ww.total.net/~pango	544
	13	13	Root.Arts.Artists.Celebrity Artists	http://www.georgeclintonart.com	1242
	14	14	Root.Arts.Artists.Celebrity Artists	http://www.tollercranston.com	2082
	15	15	Root.Arts.Artists.Celebrity Artists	http://www.tonycurtis.com/index.jsp	3370
	16	16	Root.Arts.Artists.Ceramics	http://www.claytonbailey.com	4850
	17	17	Root.Arts.Artists.Ceramics	http://www.acga.net/cbpotter	3481
	18	18	Root.Arts.Artists.Ceramics	http://kristina.search.bg	3167
	19	19	Root.Arts.Artists.Ceramics	http://www.suzybirstein.com	3467
	20	20	Root.Arts.Artists.Ceramics	http://www.bloomfieldpottery.com	4631
2	1	21	Root.Arts.Artists.Collaborative Projects	http://www.algonet.se/~haanmo/bitw	2560
	2	22	Root.Arts.Artists.Collaborative Projects	http://www.eatinri.com/flickers	2176
	3	23	Root.Arts.Artists.Collaborative Projects	http://www.irate.shadylady.org	615
	4	24	Root.Arts.Artists.Collage and Assemblage	http://www.cklebanoff.com	9143
	5	25	Root.Arts.Artists.Collage and Assemblage	http://www.laudator.com	14665
	6	26	Root.Arts.Artists.Collage and Assemblage	http://www.danlevin.com	17263
	7	27	Root.Arts.Artists.Collage and Assemblage	http://www.patstreet.com	6873
	8	28	Root.Arts.Art History	http://www.nd.edu/~crosenbe/jobs.html	50166
	9	29	Root.Arts.Art History	http://www.pbs.org/wnet/americanvisions	6421
	10	30	Root.Arts.Artists.Collage and Assemblage	http://www.ainet.or.jp/~ryuta	1497
3	1	31	Root.Arts.Artists.Collage and Assemblage	http://www.art-collages.com	1037
	2	32	Root.Arts.Artists.Collage and Assemblage	http://www.crystaltower.com/jeffw	2883
	3	33	Root.Arts.Artists.Collage and Assemblage	http://www.lobue-art.com	3145
	4	34	Root.Arts.Artists.Collectives	http://adaweb.walkerart.org	1547
	5	35	Root.Arts.Artists.Collectives	http://www.artspace111.com	2237
	6	36	Root.Arts.Artists.Collectives	http://www.arfarfarf.com	9126
4	1	37	Root.Arts.Artists.Collectives	http://www.sttf.org/	14656
	2	38	Root.Arts.Artists.Collectives	http://ds.dial.pipex.com/katy.mckay/sited	612
	3	39	Root.Arts.Artists.Collectives	http://home.btclick.com/suncafe	10592
	4	40	Root.Arts.Artists.Collectives	http://www.zukunftsmusik.com	13228
5	1	41	Root.Arts.Organizations.Art History	http://www.medievalart.org	7520
	2	42	Root.Arts.Organizations.Councils	http://www.artpridenj.com	1109
6	1	43	Root.Arts.Organizations.Councils	http://www.kazooart.org/	18017

5.3.5 Obtenção de uma seqüência de acessos a objetos do Universo

A seqüência de acessos aleatórios a objetos do Universo é obtida através da aplicação do Método de Monte Carlo, a fim de se respeitar a distribuição dos objetos em categorias de acessos. A aplicação do método será ilustrada a seguir, utilizando a Tabela Universo do Exemplo 6.

Primeiramente, é construída a Tabela 5.1 que contém todos os dados necessários para aplicação do Método de Monte Carlo. Cada linha da tabela corresponde a uma categoria ***c*** de acessos. Para cada categoria constam a quantidade ***q*** de objetos, a probabilidade relativa da categoria e o correspondente intervalo entre zero e um. A probabilidade relativa de uma categoria é obtida pela divisão da quantidade de objetos da categoria pelo total de objetos:

$$P_c = \frac{n_c}{\sum_c n_c}$$

O limite inferior correspondente a uma categoria *c* é aberto, exceto se for zero, e dado pela somatória das probabilidades relativas das categorias anteriores:

$$\sum_{i=1}^{c-1} Pi$$

O limite superior correspondente a uma categoria ***c*** é fechado, exceto se for um, e dado pela somatória da probabilidade relativa da categoria ***c*** com as probabilidades das categorias seguintes:

$$\sum_{i=1}^{c} Pi$$

Uma vez construída a tabela de probabilidades das categorias, faz-se a escolha aleatória de objetos que definirão a seqüência de acessos. A Tabela 5.2 ilustra uma seqüência de 28 acessos obtidos dessa forma. A obtenção de cada um dos acessos ocorre da seguinte forma: (i) escolha aleatória de d_1, tal que $0 < d1 \leq 1$; (ii) mapeamento de d_1 para o intervalo correspondente da Tabela 5.1, determinando a categoria ***c*** escolhida; (iii) determinação da quantidade de candidatos válidos V_c para a categoria ***c;*** (iv) escolha aleatória de um candidato d_2 da categoria ***c*** tal que $1 \leq d2 \leq vc$; (v) mapeamento do candidato d_2 escolhido para o objeto

correspondente, representados na tabela pelo seu **id**. A escolha desse objeto deve ser contabilizada e, se o número de acessos a esse objeto atingir o seu limite (o próprio número da categoria), o objeto deve ser marcado como inválido para escolha dentro da sua categoria.

Por exemplo, no passo 8 ocorrem os seguintes passos: (i) escolha aleatória de d_1 = 0,66; (ii) mapeamento de 0,66 para a categoria c = 2, pois 0,47 < 0,66 <= 0,70; (iii) determinação da quantidade de candidatos válidos V_c = 10, para a categoria 2; (iv) escolha aleatória, entre 1 e 10, do candidato d_2 = 2, da categoria 2; (v) mapeamento do candidato 2 da categoria 2 de acordo com a Tabela Universo, obtendo o objeto com **id** = 22. Como conseqüência o contador de escolhas desse objeto é incrementado de 1 para 2 (a primeira escolha desse objeto ocorreu no passo 7), o seu limite máximo, tornando esse objeto inválido na categoria para futuras escolhas. Portanto, o passo 8 resultará num acesso ao objeto:

Root.Arts.Artists.Collaborative Projects;http://www.eatinri.com/flickers;2176

Categoria (Quantidade de acessos por objetos) c	Quantidade de objetos por categoria q	Probabilidade $P_c = \frac{q_c}{\sum_c q_c}$	Intervalo	
			Limite inferior (aberto) $\sum_{i=1}^{c-1} Pi$	Limite superior (fechado) $\sum_{i=1}^{c} Pi$
1	20	0,47	0,00	0,47
2	10	0,23	0,47	0,70
3	6	0,14	0,70	0,84
4	4	0,09	0,84	0,93
5	2	0,05	0,93	0,98
6	1	0,02	0,98	1,00
Somatória	43	1		
	$\sum_c q_c$	$\sum_c p_c$		

Tabela 5.1: Distribuição de probabilidade para aplicação do Método de Monte Carlo

Passo	$0<d_1 \leq 1$	Categoria c	Quantidade de candidatos válidos na v_c	$1 \leq d_2 \leq v_c$	Id
1	0,73	3	6	1	31
2	0,65	2	10	5	25
2	0,85	4	4	3	39
4	0,04	1	20	15	15
5	0,75	3	6	2	32
6	0,65	2	10	3	23
7	0,57	2	10	2	22
8	0,66	2	10	2	22
9	0,92	4	4	1	37
10	0,64	2	9	6	27
11	0,62	2	9	6	27
12	0,45	1	19	6	6
13	0,81	3	6	4	34
14	0,96	5	2	1	41
15	0,38	1	18	7	8
16	0,82	3	6	3	33
17	0,86	4	4	4	40
18	0,34	1	17	4	4
19	0,62	2	8	6	28
20	0,27	1	16	2	2
21	0,25	1	15	4	7
22	0,84	3	6	2	32
23	0,30	1	14	12	18
24	0,92	4	4	4	40
25	0,28	1	13	5	10
26	0,39	1	12	5	11
27	0,67	2	8	5	26
28	0,70	2	8	5	26

Tabela 5.2: Aplicação do Método Monte Carlo para n=28 (nº de acessos na simulação)

Na aplicação do Método de Monte Carlo para a seqüência de passos para n = 28, demonstramos o estado inicial, quando ainda não fora eleita nenhuma categoria de objetos. Na aplicação do método no passo (1), a categoria de objeto escolhida foi a categoria c = 3 e dentre os candidatos válidos na categoria v_c, o eleito foi o objeto 1 com identificação 31. O contador de escolha da categoria foi incrementado em 1, pois o limite máximo na categoria é 3.

Para o passo (2), a categoria eleita foi a de número 2 e o objeto escolhido na seqüência da categoria foi o número 5. A identificação do objeto escolhido neste passo é a de 25 e o contador de acesso foi modificado para 1, estando ainda no limite para uma nova escolha dentro dos passos. No passo (3), a categoria escolhida foi a 4 e o objeto escolhido foi o 3, com identificação 39, e seu contador de escolha incrementado em 1. No passo (4), a categoria escolhida foi a 1 e a seqüência de candidatos válidos na categoria foi a de número 15. E a identificação do objeto é o 15, o contador de escolha neste passo passou a ser 1, eliminando a possibilidade de nova escolha pelo Método Monte Carlo, pois esse era o limite máximo do objeto na categoria. O objeto escolhido foi o 15, sendo posicionado no final da seqüência, na categoria do passo (5). A seqüência da categoria foi toda ela ordenada onde, antes da escolha do objeto 15, a categoria possuía 20 candidatos válidos e agora possui apenas 19 objetos válidos. Com isso, o trabalho dos próximos passos fica trivial, bastando seguir os passos até o seu estado final, (Ver Anexo V).

Nos passos para n = 28 não aconteceu, em nenhum momento, o esgotamento dos candidatos válidos em nenhuma das categorias. Mas, aplicando-se o Método de Monte Carlo para n =50, conforme mostra a Tabela 5.3, verifica-se que na categoria 1 todos os objetos válidos foram esgotados, justamente no passo 50 (a Tabela 5.4(a) e a Tabela 5.4(b) mostram o seu estado inicial e o seu estado final).

Caso fosse necessário escolher mais objetos além dos 50 escolhidos, o esgotamento da categoria 1 deveria ser devidamente tratado. Basicamente há três alternativas:

1. Continuar com as escolhas, eliminando a categoria 1.
2. Zerar o contador de escolha de cada elemento da categoria 1.
3. Zerar o contador de escolha de todos os elementos de todas as categorias

Dentre essas, a única que não interfere no Método de Monte Carlo é a alternativa 3.

Passo	$0<d_1 \leq 1$	Categoria c	Qtde. de candidatos válidos na categoria v_c	$1 \leq d_2 \leq v_c$	Id
1	0,33	1	20	9	9
2	0,28	1	19	4	4
3	0,86	4	4	1	37
4	1,00	6	1	1	43
5	0,64	2	10	1	21
6	0,40	1	18	11	13
7	0,68	2	10	1	21
8	0,20	1	17	7	8
9	0,20	1	16	12	16
10	0,30	1	15	2	2
11	0,56	2	9	2	23
12	0,94	5	2	2	42
13	0,42	1	14	8	12
14	0,88	4	4	1	37
15	0,87	4	4	3	39
16	0,92	4	4	2	38
17	0,34	1	13	7	11
18	0,02	1	12	6	10
19	0,05	1	11	10	19
20	0,50	2	9	1	22
21	0,05	1	10	4	6
22	0,91	4	4	3	39
23	0,61	2	9	9	30
24	0,67	2	9	6	27
25	0,63	2	9	9	30
26	0,12	1	9	2	3
27	0,30	1	8	5	15
28	0,54	2	8	4	25
29	0,64	2	8	8	29
30	0,25	1	7	1	1
31	0,92	4	4	2	38
32	0,79	3	6	1	31
33	0,04	1	6	3	14
34	0,22	1	5	1	5
35	0,99	6	1	1	43
36	0,77	3	6	4	34
37	0,53	2	8	6	27
38	0,22	1	4	2	17
39	0,95	5	2	1	41

Passo	$0<d_1\leq 1$	Categoria c	Qtde. de candidatos válidos na categoria v_c	$1\leq d_2\leq v_c$	Id
40	0,70	2	7	4	25
41	0,70	2	6	6	29
42	0,21	1	3	1	7
43	0,65	2	5	4	26
44	0,92	4	4	3	33
45	0,52	2	5	5	28
46	0,94	5	2	1	41
47	0,69	2	5	4	26
48	0,69	2	4	3	24
49	0,11	1	2	2	20
50	0,03	1	1	1	18

Tabela 5.3: Aplicação do Método de Monte Carlo para n=50 (nº de acessos na simulação)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	15	0
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Tabela 5.4 (a) Estado inicial para n = 50

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	X	18	1
	X	20	1
	X	7	1
	X	17	1
	X	5	1
	X	14	1
	X	1	1
	X	15	1
	X	3	1
	X	6	1
	X	19	1
	X	10	1
	X	11	1
	X	12	1
	X	2	1
	X	16	1
	X	8	1
	X	13	1
	X	4	1
	X	9	1
2	1	22	1
	2	23	1
	3	24	1
	4	28	1
	X	26	2
	X	29	2
	X	25	2
	X	27	2
	X	30	2
	X	21	2
3	1	31	1
	2	32	0
	3	33	0
	4	34	1
	5	35	0
	6	36	0
4	1	37	2
	2	38	2
	3	39	3
	4	40	0
5	1	41	2
	2	42	1
6	1	43	2

Tabela 5.4 (b) Estado final para n = 50

5.4 Simulação de estratégias de substituição de objetos em cache

O processo de preparação de dados para simulação de cache permitiu a geração de dados para a simulação da estratégia proposta nesta dissertação, a estratégia LSR, assim como a simulação, para fins de comparação, das principais estratégias atualmente empregadas na Internet, nominalmente, as estratégias *SIZE*, LFU e LRU. Nesta seção, são apresentados os dados preparados para a simulação, assim como os resultados obtidos e uma análise comparativa destes.

Primeiramente, obteve-se a distribuição de acessos a objetos verificada na prática, a partir do arquivo de registros de acessos do *proxy Squid* da PUCPR, por um certo período, totalizando 2.000.000 de acessos a 866.915 objetos (Filtro A e Filtro B). O Gráfico 1 contém um histograma com a distribuição verificada.

Na seqüência, foi extraído do *site* do *Yahoo*! um subconjunto com 983 objetos, definindo, assim, um Universo de Objetos e o correspondente conjunto Caminhos, com 180 entradas (Filtro C). A distribuição de acessos do *Squid* foi reproduzida sobre o Universo de Objetos (Filtro D). O Gráfico 2 contém um histograma com a distribuição obtida. Observa-se a correspondência entre o histograma obtido e o histograma mostrado no Gráfico 1, demonstrando que a distribuição dos objetos do Universo em categorias de acesso segue a mesma distribuição que se verificou na prática.

Gráfico 1: Distribuição de acessos para os objetos do Squid

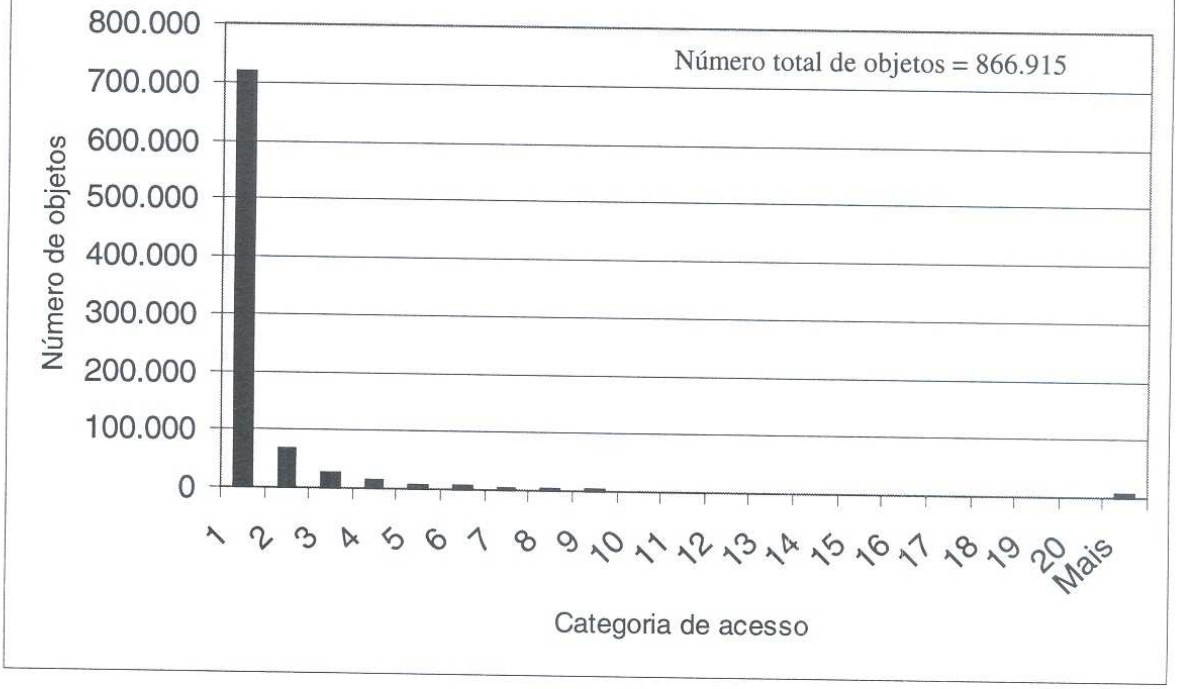
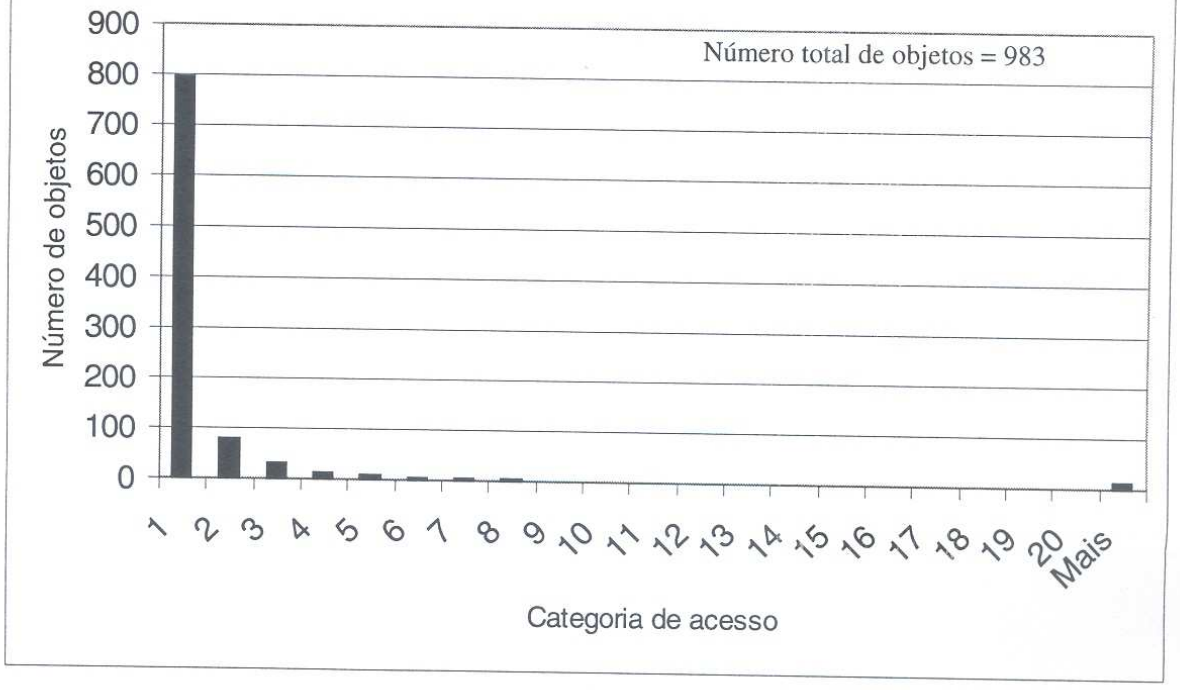
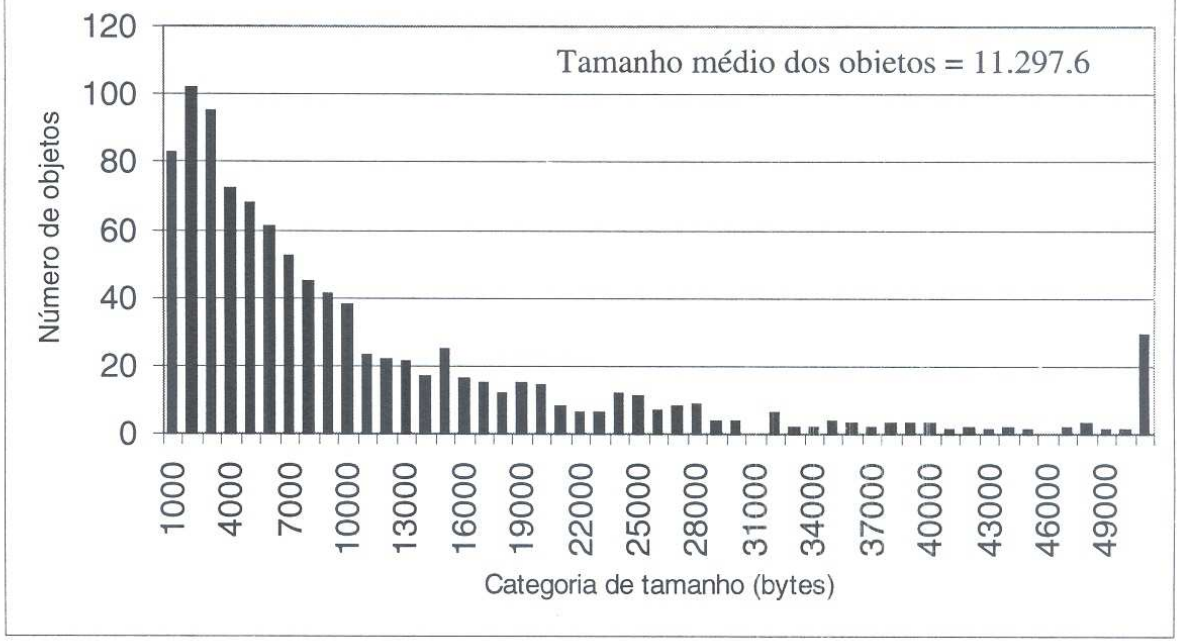


Gráfico 2: Distribuição de acessos para os objetos do Universo



O Gráfico 3 mostra um histograma com a distribuição dos objetos do Universo de acordo com os seus tamanhos, em bytes. A escolha aleatória dos objetos, a partir do *site* do *Yahoo*!, gerou um conjunto de objetos cuja distribuição por tamanho mostra a tendência de se ter muitos objetos de tamanho pequeno e poucos objetos de tamanho grande, relativamente.

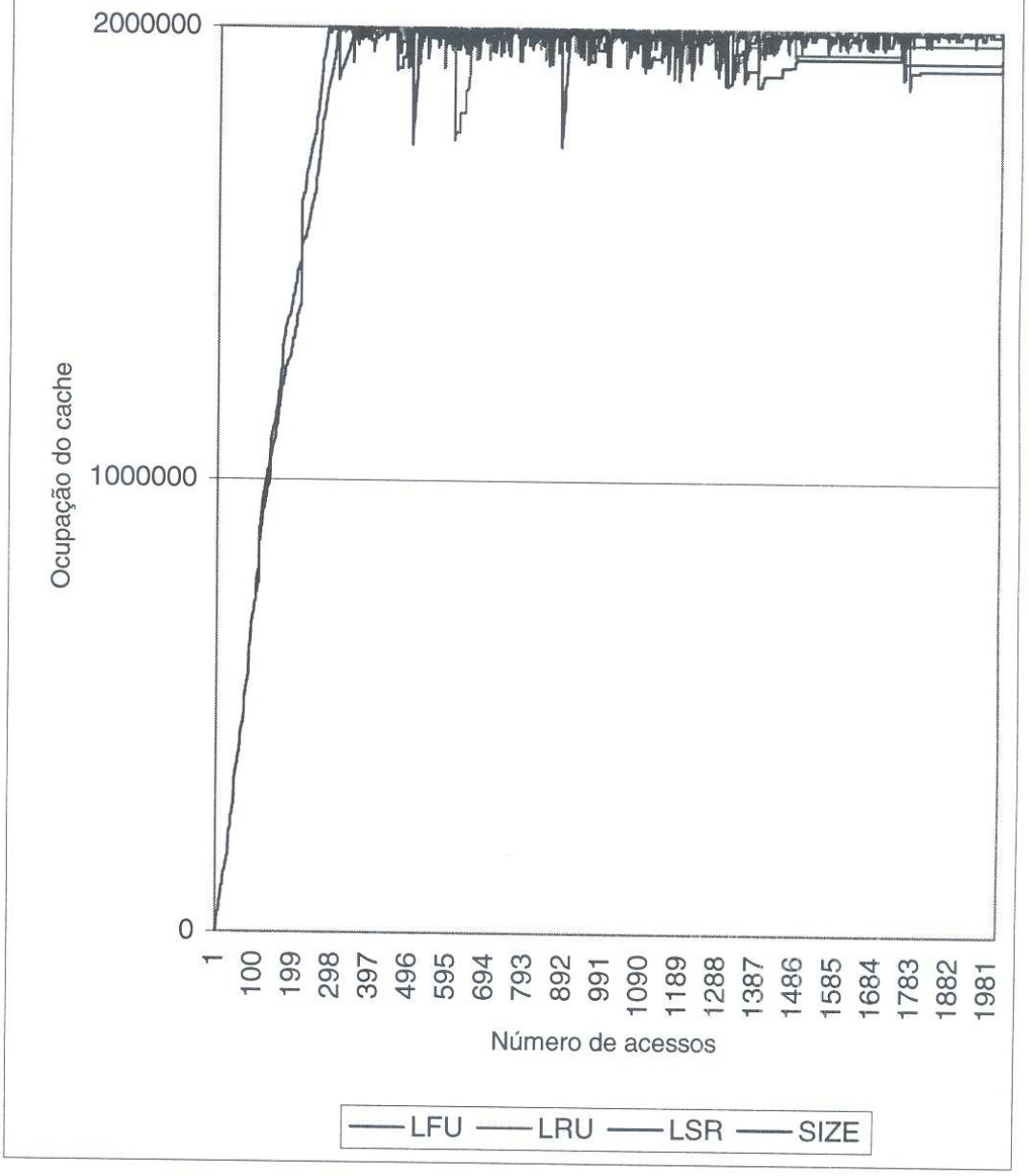
Gráfico 3: Distribuição dos objetos do Universo de acordo com os seus tamanhos



Uma vez obtida a distribuição de acessos para os objetos do Universo, foi aplicado o Filtro E para se obter a correspondente Tabela Universo, possibilitando, assim, a aplicação do Método de Monte Carlo (Filtro F) para a obtenção de seqüências de acessos. Foram geradas seqüências para 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000, 10.000, 15.000, 20.000 e 30.000 acessos. Essas 13 seqüências foram submetidas para as quatro estratégias (LSR, SIZE, LRU e LFU) em análise, para os seis seguintes tamanhos de cache: 1.000.000, 1.5000.000, 2.000.000, 2.5000.000, 5.000.000 e 10.000.000. Portanto, foram realizadas 4 x 13 x 6 = 312 simulações de cache. Cada uma das seqüências sofreu um rearranjo na ordem dos acessos a fim de contemplar o comportamento suposto dos usuários que acessam objetos na Internet, isto é, que um usuário permanece um "certo tempo" pesquisando um mesmo assunto antes de passar a pesquisar outro. Tal rearranjo foi realizado com a ordenação dos acessos de acordo com o primeiro nível de assuntos nos caminhos dos acessos. Os principais resultados destas simulações são apresentados a seguir.

O Gráfico 4 mostra a ocupação do cache para as quatro estratégias, considerando uma combinação em particular: cache de tamanho 2.000.000 e seqüência de 2.000 acessos. Observa-se que todas as estratégias tendem a manter o cache o mais ocupado possível, o que demonstra que as implementações dessas estratégias estão corretas neste sentido. (Todas as demais combinações permitiriam essa mesma observação e, por isso, não são aqui representadas.)

Gráfico 4: Ocupação de um cache de tamanho 2.000.000
em uma seqüência de 2.000 acessos a objetos



Os gráficos de 5 a 10 mostram o efeito do número de objetos acessados em uma seqüência na probabilidade de acerto. Cada gráfico corresponde aos experimentos realizados para as quatro estratégias, considerando um certo tamanho de cache. Observa-se que, em todas as situações, a probabilidade de acerto para todas as estratégias aumenta, tendendo ao máximo (ou seja, 1.0), a medida que aumenta o número de acessos a objetos. Pode-se observar também que, para as quatro estratégias, a probabilidade de acerto aumenta a medida que aumenta o tamanho do cache. Essa constatação pode ser mais facilmente notada nos gráficos de 11 a 13, que mostram o efeito do tamanho do cache na probabilidade de acerto para um determinado número de acessos. Os gráficos de 11 a 13 permitem ainda observar que há uma convergência na probabilidade de acerto entre as quatro estratégias a medida que o cache aumenta, isto é, a estratégia adotada não influencia na probabilidade de acerto. A explicação para isso é que, a medida que o cache aumenta, mais chance tem de conter todo o Universo de Objetos.

Gráfico 5: Efeito do número de objetos acessados em uma seqüência na probabilidade de acerto para um cache de tamanho 1.000.000

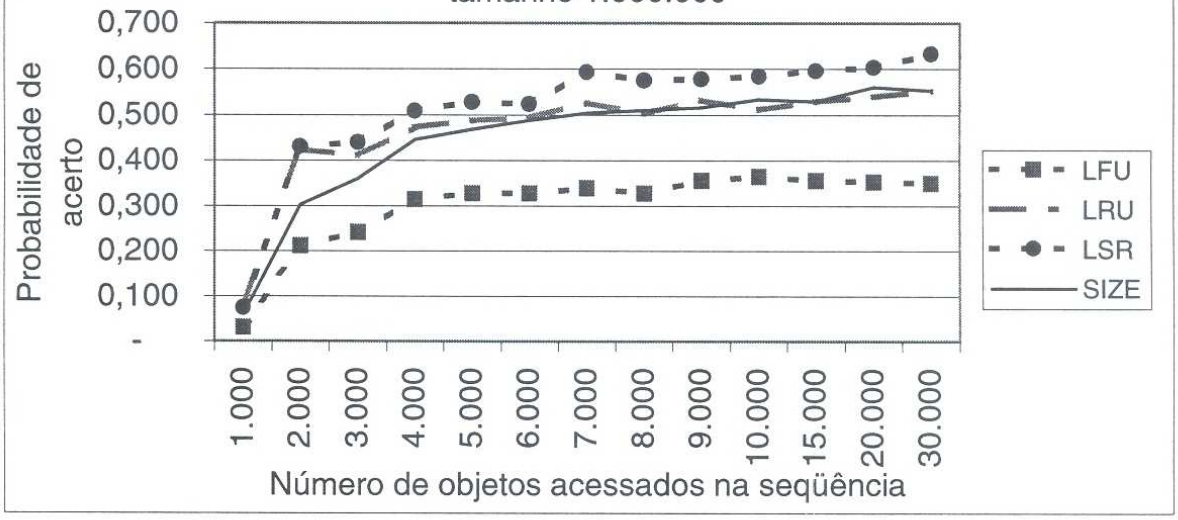


Gráfico 6: Efeito do tamanho da seqüência de acessos na probabilidade de acerto para um cache de tamanho 1.500.000

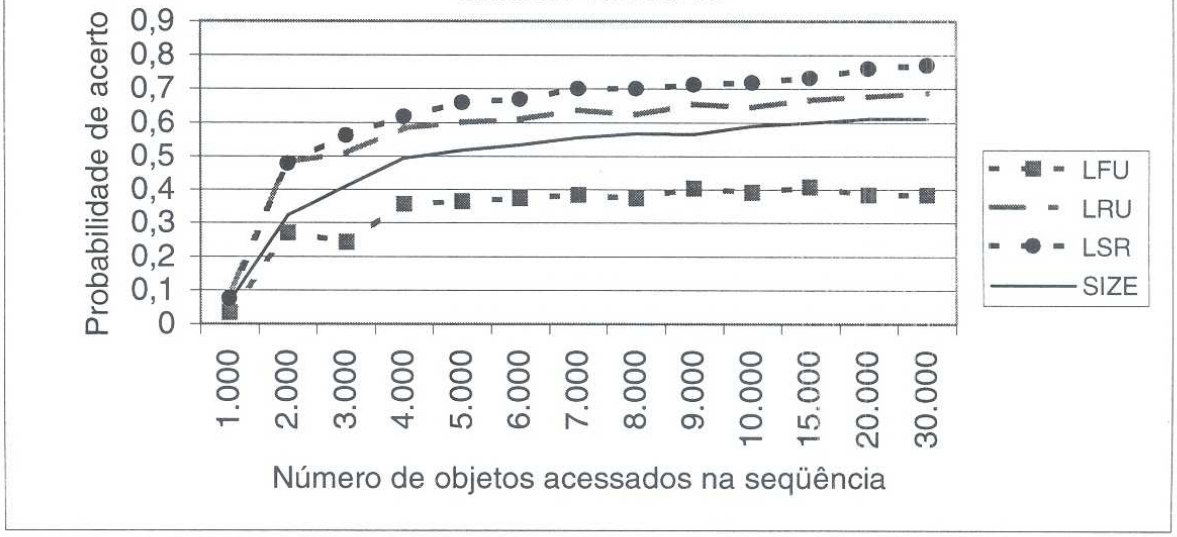


Gráfico 7: Efeito do número de objetos acessados em uma seqüência na probabilidade de acerto para um cache de tamanho 2.000.000

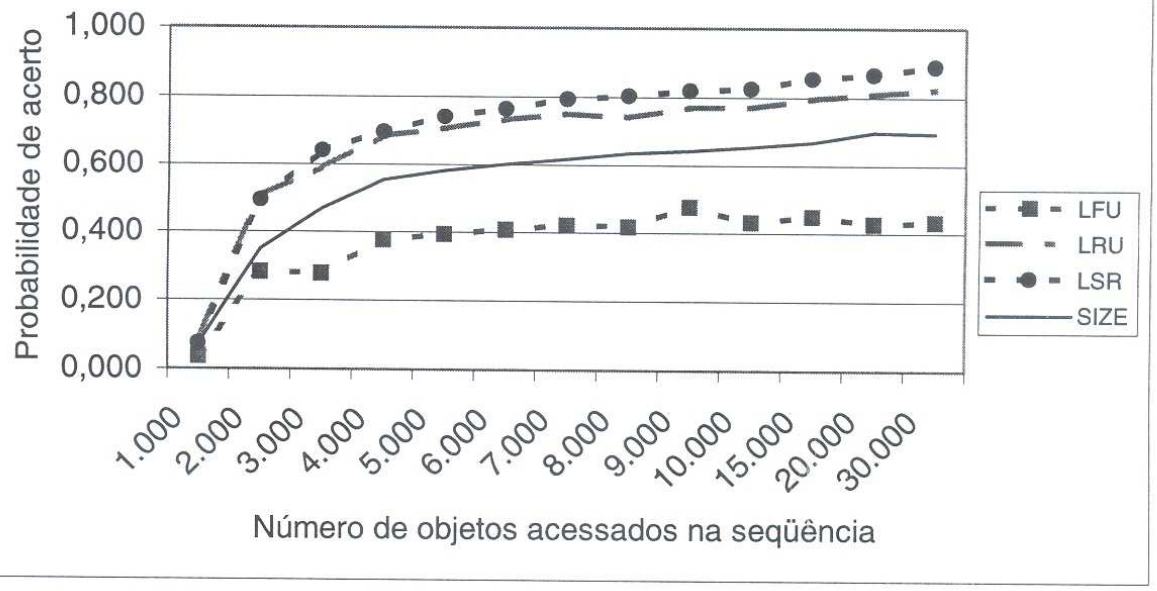


Gráfico 8: Efeito do número de objetos acessados em uma seqüência na probabilidade de acerto para um cache de tamanho 2.500.000

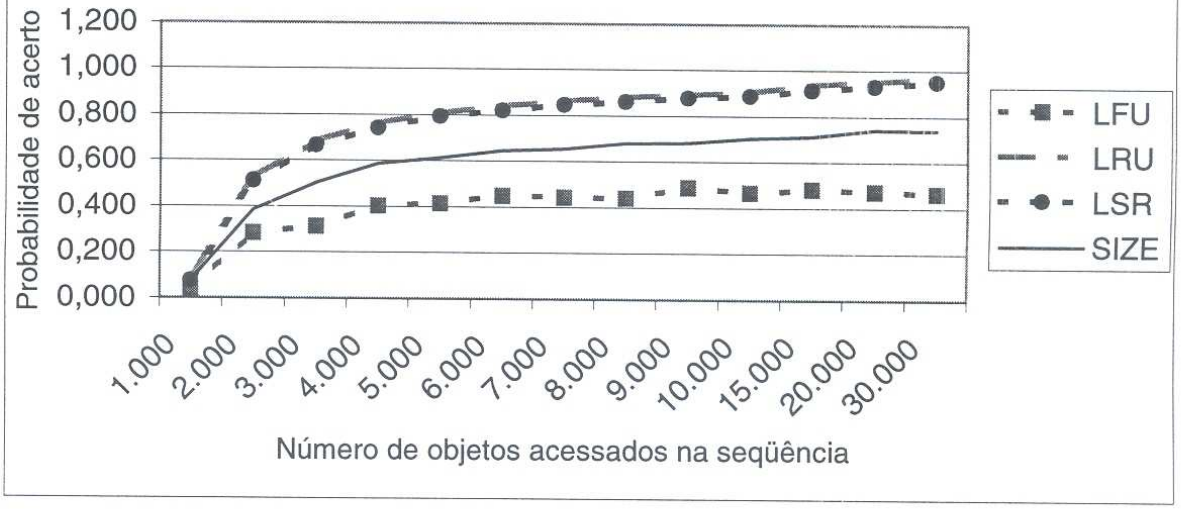


Gráfico 9: Efeito do número de objetos acessados em uma seqüência na probabilidade de acerto para um cache de tamanho 5.000.000

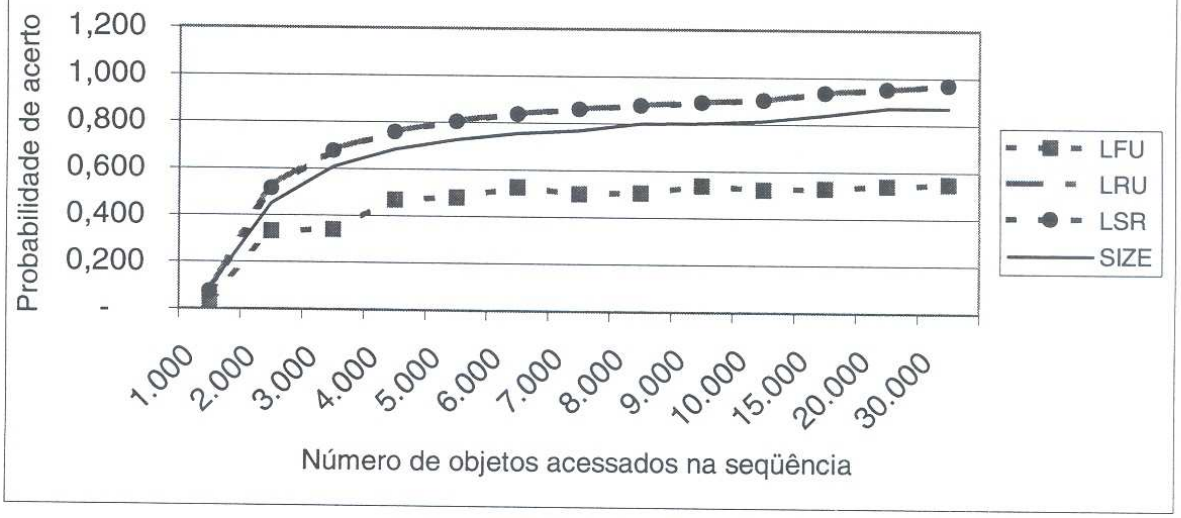


Gráfico 10: Efeito do número de objetos acessados em uma seqüência na probabilidade de acerto para um cache de tamanho 10.000.000

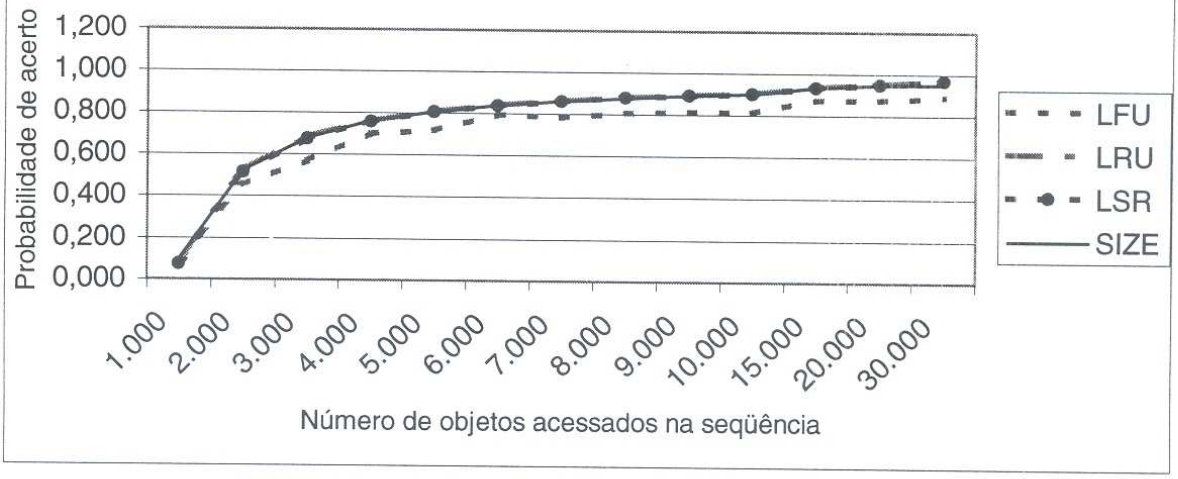


Gráfico 11: Efeito do tamanho do cache na probabilidade de acerto para 1.000 acessos

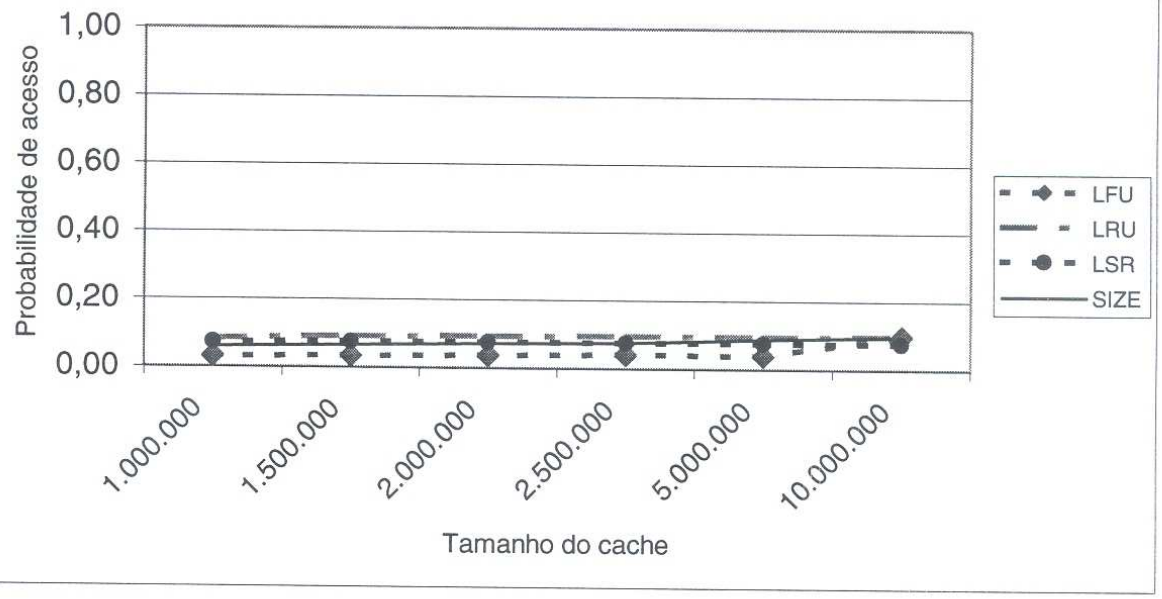
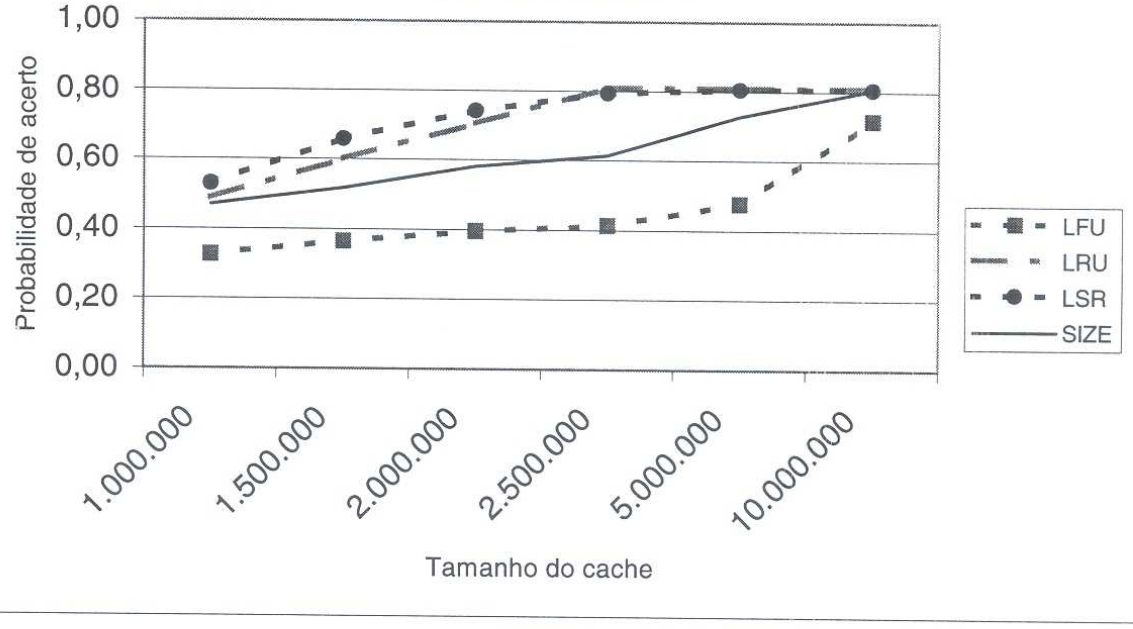
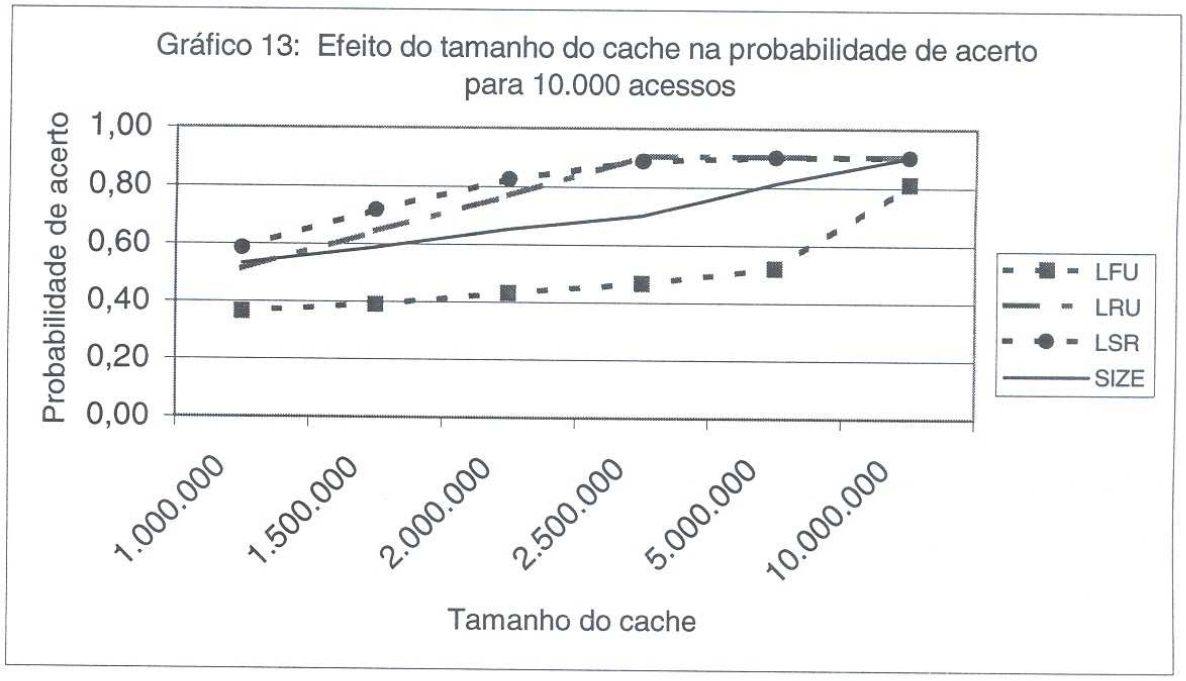


Gráfico 12: Efeito do tamanho do cache na probabilidade de acerto para 5.000 acessos





A observação final e mais importante é que, praticamente em todas as situações, verifica-se que a estratégia LSR tem maior probabilidade de acerto que as demais estratégias experimentadas. Não foi feita uma análise dos dados a fim de se determinar quanto maior é a probabilidade de acerto da estratégia LSR em relação às outras estratégias, visto que a amostragem de dados utilizada para a simulação ainda deve ser ampliada para dar maior precisão ao experimento.

CAPÍTULO 6

Conclusões e trabalhos futuros

6.1 Conclusões e propostas de trabalhos futuros

Esta dissertação propõe uma solução inovadora para o tratamento de caches na *Web* com relação à substituição de objetos: a estratégia LSR, que utiliza a classificação dos objetos para dar prioridade de permanência no cache aos objetos mais relacionados semanticamente com o objeto que se deseja inserir no cache. Um algoritmo e uma estrutura de dados para a estratégia LSR foram completamente projetados e implementados para fins de validação e comparação com outras estratégias de substituição de objetos em cache; foram também implementadas as estratégias SIZE, LFU e LRU. Além disso, foi projetado um modelo para a realização dos experimentos, incluindo toda a coleta e preparação de dados para análise. Os resultados experimentais mostraram que a estratégia proposta oferece, na maioria das situações, uma medida de eficiência – probabilidade de acerto – superior às outras estratégias. Esses indicadores favoráveis, justificam a continuidade do trabalho apresentado; abaixo seguem alguns trabalhos futuros.

(1) O desenvolvimento da estratégia LSR levou em consideração a disponibilização da semântica da informação contida em cada objeto. Com a adoção de XML, isso tende a tornar-se factível. Assim, uma direção para pesquisas futuras é explorar esse padrão, inclusive na organização da árvore semântica com a qual pode-se considerar estabelecer uma classificação padrão de objetos de informação. Certamente isso possibilitará a aplicação da estratégia LSR em sistemas reais, tornando possível uma análise sua em um ambiente mais favorável. Uma conseqüência direta da aplicação da estratégia LSR em sistemas reais é a sua implementação em *browsers*, *proxies* e servidores, o que exigirá mais esforço de desenvolvimento e pesquisas.

(2) Uma das melhorias que podem ser feitas na estratégia LSR é criar nós de assuntos na árvore de semântica, de forma dinâmica, a fim de refletir novos assuntos de interesse para os clientes e, assim, automaticamente aumentar a probabilidade de acertos do cache. Um sinal de que a árvore precisa ser revista, isto é, que novos assuntos devem ser considerados, é quando os nós *etc* se encontram com muitos objetos de informação associados. Seria possível,

ainda, controlar o cache do cliente para saber o perfil do cliente nas consultas e, a partir daí, dinamicamente, configurar a árvore de assuntos para o cliente.

(3) Outra melhoria é evitar remover um objeto eleito como o mais distante semanticamente mas que tenha alto valor (por exemplo, uso freqüente, alto custo para *download*, etc) para outro cliente – o que viabilizaria a aplicação da estratégia LSR para caches multi-usuários – ou outra ``*thread*'' de consultas do mesmo cliente, supondo que um mesmo cliente possa estar consultando objetos sobre mais de um assunto simultaneamente.

(4) Uma questão importante a ser respondida é como a estrutura da árvore de assuntos afeta o desempenho do cache, pois a estrutura em árvore pode ser custosa ou vantajosa para se inserir e remover objetos, quando se compara com o desempenho obtido por estratégias que utilizam estruturas lineares.

(5) O algoritmo projetado e experimentado para a estratégia LSR considera, para efeito de escolher os objetos menos relacionados semanticamente, somente o objeto que se deseja inserir no cache. Uma melhoria imediata é a armazenagem do histórico das semânticas mais recentemente acessadas e utilizar tal histórico para decidir quais os objetos menos relacionados semanticamente não apenas com o último acessado, mas considerando uma janela no passado de acessos. Certamente, essa modificação deve favorecer ainda mais a estratégia LSR em sua comparação com as demais.

(6) O experimento e validação da estratégia LSR utilizou uma amostragem de dados relativamente modesta: é grande em termos absolutos, mas muito limitada, dado o volume de dados na Web. Portanto, os experimentos de comparação com outras estratégias devem ser novamente realizados para grandes massas de dados e com bastante variação de tipos de usuários e *sites*. A grande dificuldade neste processo, certamente, será em obter a classificação semântica dos objetos de informação, que depende de outro item apontado como trabalho futuro.

Portanto, as principais contribuições da dissertação são a proposta da nova estratégia de substituição de objetos em cache, a sua implementação e validação, assim como o próprio modelo e correspondentes ferramentas (chamados de filtros) de obtenção de dados para a validação. Os resultados iniciais de medição de eficiência são favoráveis à estratégia LSR e percebe-se que as perspectivas de aplicações e trabalhos futuros de pesquisa e desenvolvimento são muitas.

Referências bibliográficas

[AGG99] Aggarwal, C., Wolf, J. L., and Yu, P.S. (1999). *Caching on the World Wide Web, IEEE Computer, 11(1):94-105.*

[BES95] Bestavros, A., Carter, R. L., Crovella, M. E., Cunha, C. R., Heddaya, A., and Mirdad, S. A. Application-level document caching in the Internet. In *Proceedings of the 2nd International Workshop in Distributed and Networked Environments (IEEE SDNE '95)* (Whistler, British Columbia, June 1995). http://cs-www.bu.edu/faculty/best/res/papers/sdne95.ps.

[CAO97] Cao, P., and Irani, S. Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)* (Monterey, CA, Dec. 1997). http://www.cs.wisc.edu/~cao/papers/gd-size.ps.Z.

[CRO96] Crovella, M., and Bestavros, A. Self-similarity in World-Wide Web traffic evidence and possible causes. In *Proceedings of the SIGMETRICS '96 conference* (May 1996). http://cs-www.bu.edu/faculty/best/res/papers/sigmetrics96.ps.

[CUN95] Cunha, C. R., Bestavros, A., and Crovella, M. E. Characteristics of WWW client-based traces. Tech. Rep. BU-CS-95-010, Computer Science Department, Boston University, 111 Cummington St, Boston, MA 02215, July 1995. http://www.cs.bu.edu/techreports/95-010-www-client-traces.ps.Z.

[DIL99] Dilley, J., Arlitt, M., and Perret, S. Enhancement and validation of the Squid cache replacement policy. In *Proceedings of the 4th International Web Caching Workshop* (Apr. 1999). http://www.ircache.net/Cache/Workshop99/Papers/dilley-0.ps.gz.

[FAN98] Fan, L., Cao, P., Almeida, J., and Broder, A. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of the ACM SIGCOMM'98 conference* (Sept. 1998), pp. 254-265. http://www.cs.wisc.edu/~cao/papers/summarycache.html.

[JAC98] Jacobson, Q., and Cao, P. Potential and limits of Web prefetching between low-bandwidth clients and proxies. In *Proceedings of the Third International WWW Caching Workshop* (June 1998). http://wwwcache.ja.net/events/workshop/28/cao-prepush.ps.

[RAM94] Ramakrisna Karedla, J. Spencer Love e Bradley Wherry. Caching Strategies to Improve Disk System Performance. *IEEE Computer,* p. 38-46, March 1994

[KRI99] Krishnamurthy, B., and Wills, E. C. Proxy cache coherency and replacement-Towards a more complete picture. *In Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, Austin, TX, (June 1999).* http://www.att.com/~bala/papers/ccrcp.ps.gz

[LIG99] Light, R. (1999). *Iniciando em XML, Ed. Makron*

[LOR96] Lorenzetti, P., Rizzo, L., and Vicisano, L. (1996) Replacement Policies for a proxy cache. URL: http://www.iet.unipi.it/luigi/caching.ps.gz

[EVA96] Evangelos P. Markatos. Main Memory Caching of Web Documents. In *Proceedings of the Fifth International WWW Conference*, 1996.

[MUR98] Murta, C. D., Almeida, V. A. F., and Jr., W. M. Analyzing performance of partitioned caches for the WWW. In *Proceedings of the 3rd International WWW Caching Workshop* (June 1998). http://wwwcache.ja.net/events/workshop/24/.

[NIC98] Nicolas Niclause, Zhen Liu e Philippe Nain. A New and Efficient Caching Policy for the World Wid Web. In *Proceedings of the 1998 Workshop on Internet Server Performance*, June 1998.

[NEI93] O'Neil, E,. O'Neil, P., and Weikum, G. (1993). The LRU-K page replacement algorithm for database disk buffering. *In Proceedings of the ACM SIGMOD International Conference on Management of Data.*

[PIT94] Pitkow, J. E., and Recker, M. M. A simple yet robust caching algorithm based on dynamic access patterns. In *Proceedings of the 3rd International WWW Conference* (Chicago, Oct. 1994). http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/pitkow/caching.html.

[RIZ98] Rizzo, L., and Vicisano, L. Replacement policies for a proxy cache. Tech. Rep. RN/98/13, UCL-CS, 1998. http://www.iet.unipi.it/~luigi/lrv98.ps.gz.

[ROD99] Rodriguez, P., Spanner, C., and Biersack, E. W. Web caching architectures: Hierarchical and distributed caching. In *Proceedings of the 4th International Web Caching Workshop* (Apr. 1999). http://www.ircache.net/Cache/Workshop99/Papers/rodriguez-final.ps.gz.

[ROD99] Rodriguez, P., Spanner, C., and Biersack, E. W. Web caching architectures: Hierarchical and distributed caching. In *Proceedings of the 4th International Web Caching Workshop* (Apr. 1999). http://www.ircache.net/Cache/Workshop99/Papers/rodriguez-final.ps.gz.

[SCH97] Scheuermann, P., Shim, J., and Vingralek, R. A case for delay-conscious caching of Web documents. In *Proceedings of the 6th International WWW Conference* (Santa Clara, Apr. 1997). http://www.scope.gmd.de/info/www6/technical/paper020/paper20.html.

[SIL94] Silberschatz, A. and Galvin, P. B. (1994). Operating Systems Concepts. Addison-Wesley, Reading, Mass., fourth edition.

[WES95] Wessels, D. Intelligent caching for World-Wide Web objects. In *Proceedings of the INET '95 conference* (Honolulu, Hawaï, June 1995). http://www.nlanr.net/~wessels/Papers/wessels-inet95/wessels-inet95.ps.gz.

[WIL96] Williams, S., Abrams, M., Standridge, C. R., Abdulla, G., and Fox, E. A. Removal policies in network caches for World-Wide Web documents. In *Procedings of the ACM SIGCOMM '96 Conference* (Stanford University, CA, Aug. 1996). http://ei.cs.vt.edu/~succeed/96sigcomm/.

[WOO97] Wooster, R. P., and Abrams, M. Proxy caching that estimate page load delays. In *Proceedings of the 6rd International WWW Conference* (Apr. 1997). http://www.scope.gmd.de/info/www6/technical/paper250/paper250.html.

[WOR94] Worrell, K. J. Invalidation in large scale network objects caches. Master's thesis, Faculty of the graduate school of the University of Colorado, 1994. ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/WorrellThesis.ps.Z.

[YOU94] Young, N. (1994). The K-server dual loose competitive for paging. *Algorithmica*, 11,(6):525-541. Rewritten version of online caching as cache size varies, in The 2nd Annual ACM-SIAM Symposium on Discrete Algorithms, pages 241-250, 1991.

[CHI99] Chidlovskii, B., Roncancio, C and Schneider M. L. (1999). Semantic cache mechanism for heterogeneous web querying; 1-23. http://www8.org/w8-papers/3a-search-query/semantic/semantic.html

ANEXO I:

Tabela comparativa de algoritmos de substituição

Algorítmo / Características Físicas	(1) FIFO	(2) SIZE	GD-SIZE	(3) LRU	(4) LFU In-cache Perfect	Lowest Latency First Minimiza Latência Média	Estimativa de utilidade do objeto		Hyper -G	Pitkow/Recker	LOG 2–SIZE	LRU – Threshold (objetos maiores que um certo tamanho, nunca são removidos)	PSS Pyramidal Selection Scheme	Least Normalized Cost Replacement LNC – R (f)	Partioned Caches	Size Adjustment LRU (f) (SLRU)	LRUMIN
							LRV (f)	Hibrid (f) (reduz latência total)									
Instante de entrada	X																
Tamanho do objeto		X (key1)	X (key1)				X	X	X (key1)	X (key2)	X (key1)	X (condição)	X (key1)	X	X	X	X (key1)
Freqüência de uso					X									X			
Latência no Acesso			X (key2)			X											
Custo na transferência														X			
Tempo desde o último uso		X (key2)		X			X		X (key2)	X (key1)	X (key2)	X (key1)	X (key2)			X	X (key2)
Custo no acesso (rede +processador +disco)							X	X									
Time-to-Live (TTL)																	

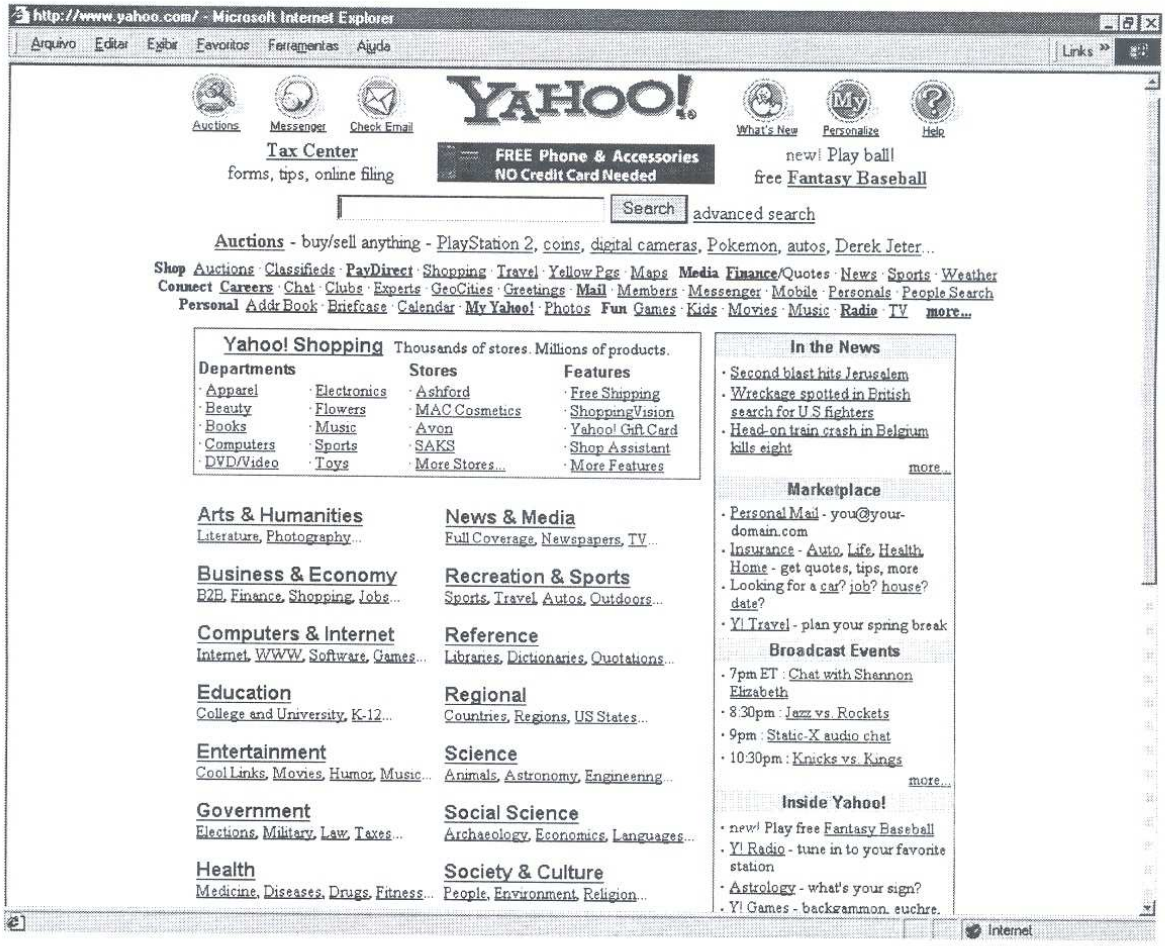
ANEXO II

Principais *sites* de busca na Web

- *AltaVista* http://www.altavista.com
- *Cade* http://www.cade.com.br
- *Excite* http://excite.com/
- *InfoSeek* http://www.infoseek.com
- *Lycos* http://www.lycos.com/
- *WebCrawlwr* http://www.webcrawler.com
- *Yahoo* http://www.yahoo.com

ANEXO III

Interface de busca do *site Yahoo*!



ANEXO IV

Seqüência de objetos acessados pelo *Proxy*

Anexo IV.1: Seqüência de objetos acessados pelo *Squid*

http://ebusiness.ittoolbox.com
http://www.net-profit-center.net
http://www.classearch.com
http://www.dci.com
http://www.sonicstate.com
http://www.worldskip.com
http://www.dtp.com
http://www.orientation.com/en/home.html
http://www.zdnet.com
http://www.cnet.com
http://www.geocities.com/Athens/Parthenon
http://www.sdexpo.com
http://www.perseus.tufts.edu
http://www.upei.ca/~siin
http://www.edupoint.com
http://www.finaid.org
http://www.heproc.org
http://www.submarine.freehosting.net
http://www.i-us.com
http://www.civsoc.com/index.htm
http://www.mobo.net/m2/index.htm
http://www.exn.ca
http://www.lmp.ucla.edu
http://www.nifi.org
http://www.campsnoopy.com
http://arsdigita.org/prize
http://www.ipdg.org
http://www.adultstudentcenter.com
http://www.vmusic.com/sdma
http://www.e-democracy.org/do
http://www.caddieshak.com
http://www.artery.org
http://www.medievalart.org

http://www.camelotthemepark.co.uk
http://www.katharinegibbs.com
http://ali.apple.com
http://www.bonfantegardens.com
http://www.alincom.com
http://www.azstarnet.com/~benok
http://www.bullseyeonthenet.com

http://www.graffiti.org
http://ericacve.org
http://www.irate.shadylady.org
http://www.nabe.org
http://www.artandphysics.com
http://www.well.ac.uk
http://www.gspyo.com/stolen
http://www.pipemedia.net/cartoons/index.htm
http://www.rsub.com/typographic
http://www.abt.org/dictionary
http://blackbeardsusa.com/
http://www.asap.net
http://www.camdenpark.com
http://fishart.free.fr
http://www.ou.edu/music/Hennagin/index.htm
http://www.kidlink.org
http://www.stoa.org/diotima
http://www.wintersim.org
http://www.mabe-mich.org
http://www.acsl.org/acsl
http://www.imaginet.fr/ime
http://freespace.virgin.net/b.mercer
http://www.slf.ruhr-uni-bochum.de
http://www.cs.orst.edu/~burnett/vpl.htm
http://www.cps.enel.ucalgary.ca
http://www.ctexpo.com/ctx
http://www.qa.org.au
http://humanities.uchicago.edu/cis
http://www127.pair.com/critical
http://eserver.org/theory

Anexo IV.2: Número de acessos por objetos do *Squid*, correspondente à seqüência do Anexo IV.1.

http://ebusiness.ittoolbox.com;1
http://www.net-profit-center.net;1
http://www.classearch.com;1
http://www.dci.com;1
http://www.sonicstate.com;1
http://www.worldskip.com;1
http://www.dtp.com;1
http://www.orientation.com/en/home.html;1
http://www.zdnet.com;1
http://www.cnet.com;1
http://www.geocities.com/Athens/Parthenon;1
http://www.sdexpo.com;1
http://www.perseus.tufts.edu;1
http://www.upei.ca/~siin;1
http://www.edupoint.com;1
http://www.finaid.org;1
http://www.heproc.org;1

http://www.submarine.freehosting.net;1
http://www.i-us.com;1
http://www.civsoc.com/index.htm;1
http://www.mobo.net/m2/index.htm;2
http://www.exn.ca;2
http://www.lmp.ucla.edu;2
http://www.nifi.org;2
http://www.campsnoopy.com;2
http://arsdigita.org/prize;2
http://www.ipdg.org;2
http://www.adultstudentcenter.com;2
http://www.vmusic.com/sdma;2
http://www.e-democracy.org/do;2
http://www.caddieshak.com;2
http://www.artery.org;2
http://www.medievalart.org;2
http://www.camelotthemepark.co.uk;2
http://www.katharinegibbs.com;2
http://ali.apple.com;3
http://www.bonfantegardens.com;3
http://www.alincom.com;3
http://www.azstarnet.com/~benok;3
http://www.bullseyeonthenet.com;3
http://www.graffiti.org;3
http://ericacve.org;3
http://www.irate.shadylady.org;3
http://www.nabe.org;3
http://www.artandphysics.com;3
http://www.well.ac.uk;4
http://www.gspyo.com/stolen;4
http://www.pipemedia.net/cartoons/index.htm;4
http://www.rsub.com/typographic;4
http://www.abt.org/dictionary;4
http://blackbeardsusa.com/;4
http://www.asap.net;4
http://www.camdenpark.com;4
http://fishart.free.fr;5
http://www.ou.edu/music/Hennagin/index.htm;5
http://www.kidlink.org;5
http://www.stoa.org/diotima;5
http://www.wintersim.org;5
http://www.mabe-mich.org;5
http://www.acsl.org/acsl;5
http://www.imaginet.fr/ime;6
http://freespace.virgin.net/b.mercer;6
http://www.slf.ruhr-uni-bochum.de;6
http://www.cs.orst.edu/~burnett/vpl.htm;6
http://www.cps.enel.ucalgary.ca/397;6
http://www.ctexpo.com/ctx/ ;8
http://www.qa.org.au;8
http://humanities.uchicago.edu/cis;8

http://www127.pair.com/critical;8
http://eserver.org/theory;12

Anexo IV.3: Universo de objetos

Root.Arts.Art History; http://www.nd.edu/~crosenbe/jobs.html;50166
Root.Arts.Art History;http://www.pbs.org/wnet/americanvisions/;6421
Root.Arts.Art History;http://users.hol.gr/~dilos/anistor/cover.htm;1213
Root.Arts.Art History.Architectural History;http://www.indiana.edu/~kglowack/athens/;2338
Root.Arts.Art History.Architectural History;http://library.thinkquest.org/10098/;5744
Root.Arts.Artists.Collage and Assemblage;http://www.flyingjeannie.com/;438
Root.Arts.Art History.Photography;http://www.photographymuseum.com/;27869
Root.Arts.Art History.Photography;http://www.pbs.org/ktca/americanphotography/;927
Root.Arts.Art History.Photography;http://www.primenet.com/~sos/photopage.html;3454
Root.Arts.Art History.Thematic;http://www.celtic-twilight.com/artists/;2192
Root.Arts.Art History.Thematic;http://members.aol.com/alanta98/cardart.html;11355
Root.Arts.Art History.Thematic;http://ww.total.net/~pango/;544
Root.Arts.Artists.Celebrity Artists;http://www.georgeclintonart.com/;1242
Root.Arts.Artists.Celebrity Artists;http://www.tollercranston.com/;2082;1
Root.Arts.Artists.Celebrity Artists;http://www.tonycurtis.com/index.jsp;3370
Root.Arts.Artists.Ceramics;http://www.claytonbailey.com/;4850
Root.Arts.Artists.Ceramics;http://www.acga.net/cbpotter/;3481
Root.Arts.Artists.Ceramics;http://kristina.search.bg/;3167
Root.Arts.Artists.Ceramics;http://www.suzybirstein.com/;3467
Root.Arts.Artists.Ceramics;http://www.bloomfieldpottery.com/;4631
Root.Arts.Artists.Collaborative Projects;http://www.algonet.se/~haanmo/bitw/;2560
Root.Arts.Artists.Collaborative Projects;http://www.eatinri.com/flickers/;2176
Root.Arts.Artists.Collaborative Projects;http://www.irate.shadylady.org/;615
Root.Arts.Artists.Collage and Assemblage;http://www.cklebanoff.com/;9143
Root.Arts.Artists.Collage and Assemblage;http://www.laudator.com/;14665
Root.Arts.Artists.Collage and Assemblage;http://www.danlevin.com/;17263
Root.Arts.Artists.Collage and Assemblage;http://www.patstreet.com/;6873
Root.Arts.Art History; http://www.nd.edu/~crosenbe/jobs.html;50166
Root.Arts.Art History;http://www.pbs.org/wnet/americanvisions/;6421
Root.Arts.Artists.Collage and Assemblage;http://www.ainet.or.jp/~ryuta/;1497
Root.Arts.Artists.Collage and Assemblage;http://www.art-collages.com/;1037
Root.Arts.Artists.Collage and Assemblage;http://www.crystaltower.com/jeffw/;2883
Root.Arts.Artists.Collage and Assemblage;http://www.lobue-art.com/;3145
Root.Arts.Artists.Collectives;http://adaweb.walkerart.org/;1547
Root.Arts.Artists.Collectives;http://www.artspace111.com/;2237
Root.Arts.Artists.Collectives;http://www.arfarfarf.com/;9126
Root.Arts.Artists.Collectives;http://www.sttf.org/;14656
Root.Arts.Artists.Collectives;http://ds.dial.pipex.com/katy.mckay/sited/;612
Root.Arts.Artists.Collectives;http://home.btclick.com/suncafe/;10592
Root.Arts.Artists.Collectives;http://www.zukunftsmusik.com/;13228
Root.Arts.Organizations.Art History;http://www.medievalart.org/;7520
Root.Arts.Organizations.Councils;http://www.artpridenj.com/;1109
Root.Arts.Organizations.Councils;http://www.kazooart.org/;18017

Anexo IV.4: Caminhos correspondentes ao Universo de objetos do Anexo IV.3

Root.Arts.Art History
Root.Arts.Art History.Architectural History
Root.Arts.Artists.Collage and Assemblage
Root.Arts.Art History.Photography
Root.Arts.Art History.Thematic
Root.Arts.Artists.Celebrity Artists
Root.Arts.Artists.Ceramics
Root.Arts.Artists.Collaborative Projects
Root.Arts.Artists.Collage and Assemblage
Root.Arts.Art History
Root.Arts.Artists.Collectives
Root.Arts.Organizations.Art History
Root.Arts.Organizations.Councils

Anexo IV.5: Distribuição de acessos para os objetos do Universo apresentados no Anexo IV.3, de acordo com a distribuição de acessos para os objetos do *Squid* da Figura 5.11

Root.Arts.Art History; http://www.nd.edu/~crosenbe/jobs.html;50166;1
Root.Arts.Art History;http://www.pbs.org/wnet/americanvisions/;6421;1
Root.Arts.Art History;http://users.hol.gr/~dilos/anistor/cover.htm;1213;1
Root.Arts.Art History.Architectural History;http://www.indiana.edu/~kglowack/athens/;2338;1
Root.Arts.Art History.Architectural History;http://library.thinkquest.org/10098/;5744; 1
Root.Arts.Artists.Collage and Assemblage;http://www.flyingjeannie.com/;438;1
Root.Arts.Art History.Photography;http://www.photographymuseum.com/;27869;1
Root.Arts.Art History.Photography;http://www.pbs.org/ktca/americanphotography/;927;1
Root.Arts.Art History.Photography;http://www.primenet.com/~sos/photopage.html;3454;1
Root.Arts.Art History.Thematic;http://www.celtic-twilight.com/artists/;2192;1
Root.Arts.Art History.Thematic;http://members.aol.com/alanta98/cardart.html;11355;1
Root.Arts.Art History.Thematic;http://ww.total.net/~pango/;544;1
Root.Arts.Artists.Celebrity Artists;http://www.georgeclintonart.com/;1242;1
Root.Arts.Artists.Celebrity Artists;http://www.tollercranston.com/;2082;1
Root.Arts.Artists.Celebrity Artists;http://www.tonycurtis.com/index.jsp;3370;1
Root.Arts.Artists.Ceramics;http://www.claytonbailey.com/;4850;1
Root.Arts.Artists.Ceramics;http://www.acga.net/cbpotter/;3481;1
Root.Arts.Artists.Ceramics;http://kristina.search.bg/;3167;1
Root.Arts.Artists.Ceramics;http://www.suzybirstein.com/;3467;1
Root.Arts.Artists.Ceramics;http://www.bloomfieldpottery.com/;4631;1
Root.Arts.Artists.Collaborative Projects;http://www.algonet.se/~haanmo/bitw/;2560;2
Root.Arts.Artists.Collaborative Projects;http://www.eatinri.com/flickers/;2176;2
Root.Arts.Artists.Collaborative Projects;http://www.irate.shadylady.org/;615;2
Root.Arts.Artists.Collage and Assemblage;http://www.cklebanoff.com/;9143;2
Root.Arts.Artists.Collage and Assemblage;http://www.laudator.com/;14665;2
Root.Arts.Artists.Collage and Assemblage;http://www.danlevin.com/;17263;2
Root.Arts.Artists.Collage and Assemblage;http://www.patstreet.com/;6873;2
Root.Arts.Art History; http://www.nd.edu/~crosenbe/jobs.html;50166;2
Root.Arts.Art History;http://www.pbs.org/wnet/americanvisions/;6421;2
Root.Arts.Artists.Collage and Assemblage;http://www.ainet.or.jp/~ryuta/;1497;2
Root.Arts.Artists.Collage and Assemblage;http://www.art-collages.com/;1037;3

Root.Arts.Artists.Collage and Assemblage;http://www.crystaltower.com/jeffw/;2883;3
Root.Arts.Artists.Collage and Assemblage;http://www.lobue-art.com/;3145;3
Root.Arts.Artists.Collectives;http://adaweb.walkerart.org/;1547;3
Root.Arts.Artists.Collectives;http://www.artspace111.com/;2237;3
Root.Arts.Artists.Collectives;http://www.arfarfarf.com/;9126;3
Root.Arts.Artists.Collectives;http://www.sttf.org/;14656;4
Root.Arts.Artists.Collectives;http://ds.dial.pipex.com/katy.mckay/sited/;612;4
Root.Arts.Artists.Collectives;http://home.btclick.com/suncafe/;10592;4
Root.Arts.Artists.Collectives;http://www.zukunftsmusik.com/;13228;4
Root.Arts.Organizations.Art History;http://www.medievalart.org/;7520;5
Root.Arts.Organizations.Councils;http://www.artpridenj.com/;1109;5
Root.Arts.Organizations.Councils;http://www.kazooart.org/;18017;6

Anexo IV.6: Seqüência de acessos a objetos do Universo obtida a partir da Tabela Universo do Exemplo 6

Seqüência de 28 acessos a objetos
Root.Arts.Artists.Collage and Assemblage;http://www.art-collages.com;1037
Root.Arts.Artists.Collage and Assemblage;http://www.laudator.com;14665
Root.Arts.Artists.Collectives;http://home.btclick.com/suncafe;10592
Root.Arts.Artists.Celebrity Artists;http://www.tonycurtis.com/index.jsp;3370
Root.Arts.Artists.Collage and Assemblage;http://www.crystaltower.com/jeffw;2883
Root.Arts.Artists.Collaborative Projects;http://www.irate.shadylady.org;615
Root.Arts.Artists.Collaborative Projects;http://www.eatinri.com/flickers;2176
Root.Arts.Artists.Collaborative Projects;http://www.eatinri.com/flickers;2176
Root.Arts.Artists.Collectives;http://www.sttf.org/;14656
Root.Arts.Artists.Collage and Assemblage;http://www.patstreet.com;6873
Root.Arts.Artists.Collage and Assemblage;http://www.patstreet.com;6873
Root.Arts.Artists.Collage and Assemblage;http://www.flyingjeannie.com;438
Root.Arts.Artists.Collectives;http://adaweb.walkerart.org;1547
Root.Arts.Organizations.Art History;http://www.medievalart.org;7520
Root.Arts.Art History.Photography;http://www.pbs.org/ktca/americanphotography;927
Root.Arts.Artists.Collage and Assemblage;http://www.lobue-art.com;3145
Root.Arts.Artists.Collectives;http://www.zukunftsmusik.com;13228
Root.Arts.Art History.Architectural History;http://www.indiana.edu/~kglowack/athens;2338
Root.Arts.Art History;http://www.nd.edu/~crosenbe/jobs.html;50166
Root.Arts.Art History;http://www.pbs.org/wnet/americanvisions;6421
Root.Arts.Art History.Photography;http://www.photographymuseum.com;27869
Root.Arts.Artists.Collage and Assemblage;http://www.crystaltower.com/jeffw;2883
Root.Arts.Artists.Ceramics;http://kristina.search.bg;3167
Root.Arts.Artists.Collectives;http://www.zukunftsmusik.com;13228
Root.Arts.Art History.Thematic;http://www.celtic-twilight.com/artists;2192
Root.Arts.Art History.Thematic;http://members.aol.com/alanta98/cardart.html;11355
Root.Arts.Artists.Collage and Assemblage;http://www.danlevin.com;17263
Root.Arts.Artists.Collage and Assemblage;http://www.danlevin.com;17263

ANEXO V

Seqüência de acesso a objetos resultante dos 28 passos na Aplicação do Método de Monte Carlo

Seqüência de passos para n =28

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	15	0
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Estado Inicial

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	15	0
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	**1**	**31**	**1**
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (1)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	15	0
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	**25**	**1**
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (2)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	15	0
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	**39**	**1**
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (3)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	**15**	**1**
	16	16	0
	17	17	0
	18	18	0
	19	19	0
	20	20	0
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (4)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	22	0
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	**32**	**1**
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (5)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	22	0
	3	**23**	**1**
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (6)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	**22**	**1**
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(7)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	**22**	**2**
	3	23	0
	4	24	0
	5	25	0
	6	26	0
	7	27	0
	8	28	0
	9	29	0
	10	30	0
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(8)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	27	0
	7	28	0
	8	29	0
	9	30	0
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	**1**	**37**	**1**
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(9)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	**27**	**1**
	7	28	0
	8	29	0
	9	30	0
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(10)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	6	0
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	**27**	**2**
	7	28	0
	8	29	0
	9	30	0
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(11)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	**6**	**1**
	7	7	0
	8	8	0
	9	9	0
	10	10	0
	11	11	0
	12	12	0
	13	13	0
	14	14	0
	15	16	0
	16	17	0
	17	18	0
	18	19	0
	19	20	0
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(12)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	7	0
	7	8	0
	8	9	0
	9	10	0
	10	11	0
	11	12	0
	12	13	0
	13	14	0
	14	16	0
	15	17	0
	16	18	0
	17	19	0
	18	20	0
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	**34**	**1**
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(13)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	7	0
	7	8	0
	8	9	0
	9	10	0
	10	11	0
	11	12	0
	12	13	0
	13	14	0
	14	16	0
	15	17	0
	16	18	0
	17	19	0
	18	20	0
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	**1**	**41**	**1**
	2	42	0
6	1	43	0

Passo(14)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	7	0
	7	**8**	**1**
	8	9	0
	9	10	0
	10	11	0
	11	12	0
	12	13	0
	13	14	0
	14	16	0
	15	17	0
	16	18	0
	17	19	0
	18	20	0
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(15)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	7	0
	7	9	0
	8	10	0
	9	11	0
	10	12	0
	11	13	0
	12	14	0
	13	16	0
	14	17	0
	15	18	0
	16	19	0
	17	20	0
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	**33**	**1**
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(16)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	4	0
	5	5	0
	6	7	0
	7	9	0
	8	10	0
	9	11	0
	10	12	0
	11	13	0
	12	14	0
	13	16	0
	14	17	0
	15	18	0
	16	19	0
	17	20	0
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	**4**	**40**	**1**
5	1	41	0
	2	42	0
6	1	43	0

Passo(17)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	**4**	**1**
	5	5	0
	6	7	0
	7	9	0
	8	10	0
	9	11	0
	10	12	0
	11	13	0
	12	14	0
	13	16	0
	14	17	0
	15	18	0
	16	19	0
	17	20	0
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(18)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	2	0
	3	3	0
	4	5	0
	5	7	0
	6	9	0
	7	10	0
	8	11	0
	9	12	0
	10	13	0
	11	14	0
	12	16	0
	13	17	0
	14	18	0
	15	19	0
	16	20	0
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	**28**	**1**
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo (19)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	**2**	**1**
	3	3	0
	4	5	0
	5	7	0
	6	9	0
	7	10	0
	8	11	0
	9	12	0
	10	13	0
	11	14	0
	12	16	0
	13	17	0
	14	18	0
	15	19	0
	16	20	0
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(20)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	**7**	**1**
	5	9	0
	6	10	0
	7	11	0
	8	12	0
	9	13	0
	10	14	0
	11	16	0
	12	17	0
	13	18	0
	14	19	0
	15	20	0
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(21)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	10	0
	6	11	0
	7	12	0
	8	13	0
	9	14	0
	10	16	0
	11	17	0
	12	18	0
	13	19	0
	14	20	0
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	**32**	**2**
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(22)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	10	0
	6	11	0
	7	12	0
	8	13	0
	9	14	0
	10	16	0
	11	17	0
	12	**18**	**1**
	13	19	0
	14	20	0
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(23)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	10	0
	6	11	0
	7	12	0
	8	13	0
	9	14	0
	10	16	0
	11	17	0
	12	19	0
	13	20	0
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	**40**	**2**
5	1	41	0
	2	42	0
6	1	43	0

Passo(24)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	**10**	**1**
	6	11	0
	7	12	0
	8	13	0
	9	14	0
	10	16	0
	11	17	0
	12	19	0
	13	20	0
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(25)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	**11**	**1**
	6	12	0
	7	13	0
	8	14	0
	9	16	0
	10	17	0
	11	19	0
	12	20	0
	X	10	1
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	26	0
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(26)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	12	0
	6	13	0
	7	14	0
	8	16	0
	9	17	0
	10	19	0
	11	20	0
	X	11	1
	X	10	1
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	**26**	**1**
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(27)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	12	0
	6	13	0
	7	14	0
	8	16	0
	9	17	0
	10	19	0
	11	20	0
	X	11	1
	X	10	1
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	0
	5	**26**	**2**
	6	28	0
	7	29	0
	8	30	0
	X	27	2
	X	22	2
3	1	31	0
	2	32	0
	3	33	0
	4	34	0
	5	35	0
	6	36	0
4	1	37	0
	2	38	0
	3	39	0
	4	40	0
5	1	41	0
	2	42	0
6	1	43	0

Passo(28)

Categoria	Seqüência na categoria	Id	Contador de Escolhas
1	1	1	0
	2	3	0
	3	5	0
	4	9	0
	5	12	0
	6	13	0
	7	14	0
	8	16	0
	9	17	0
	10	19	0
	11	20	0
	X	11	1
	X	10	1
	X	18	1
	X	7	1
	X	2	1
	X	4	1
	X	8	1
	X	6	1
	X	15	1
2	1	21	0
	2	23	0
	3	24	0
	4	25	1
	5	28	0
	6	29	0
	7	30	0
	x	26	2
	X	27	2
	X	22	2
3	1	31	1
	2	32	2
	3	33	1
	4	34	1
	5	35	0
	6	36	0
4	1	37	1
	2	38	0
	3	39	1
	4	40	2
5	1	41	1
	2	42	0
6	1	43	0

Estado Final