Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 28, Num. 02 (2021) 25-38

RESEARCH ARTICLE

Hash Tables for a Digital Lexicon

Tabelas Hash para um Léxico Digital

Fábio Carlos Moreno[1]*,Cinthyan Renata Sachs C. de Barbosa[2], Edio Roberto Manfio[3]

Abstract: This paper deals with the construction of digital lexicons within the scope of Natural Language Processing. Data Structures called Hash Tables have demonstrated to generate good results for Natural Language Interface for Databases and have data dispersion, response speed and programming simplicity as main features. The storage of the desired information is done by associating a key through the hashing functions that is responsible for distributing the information in this table. The objective of this paper is to present the tool called Visual TaHs that uses a sparse table to a real lexicon (Lexicon of Herbs), improving performance results of several implemented hash functions. Such structure has achieved satisfactory results in terms of speed and storage when compared to conventional databases and can work in various media, such as desktop, Web and mobile.
Keywords: Lexicon — Natural Language Processing — Hash

Resumo: Este artigo trata da construção de léxicos digitais no âmbito do Processamento de Linguagem Natural. Estruturas de Dados chamadas Tabelas Hash têm demonstrado gerar bons resultados para a Interface em Linguagem Natural para Banco de Dados e apresentam como características dispersão dos dados, velocidade de respostas e simplicidade de programação. O armazenamento das informações desejadas é feito por meio da associação de uma chave de funções hash que é responsável por distribuir as informações nessa tabela. O objetivo deste artigo é apresentar a ferramenta chamada Visual TaHs que usa uma tabela esparsa para um léxico real (Léxico das Ervas), melhorando os resultados de desempenho de diversas funções hash implementadas. Tais estruturas têm alcançado resultados satisfatórios em termos de velocidade e armazenamento quando comparadas aos bancos de dados convencionais, podendo funcionar em diversas mídias como *desktop, web* e *mobile*.
Palavras-Chave: Léxico — Processamento de Linguagem Natural — Hash

[1] *Centro de Ciências Tecnológicas, Universidade Estadual do Norte do Paraná (UENP), Jacarézinho - Paraná, Brasil*
[2] *Programa de Pós-Graduação em Ciência da Computação, Universidade Estadual de Londrina (UEL), Londrina - Paraná, Brasil*
[3] *Faculdade de Tecnologia Dep. Júlio Julinho Marcondes de Moura (FATEC), Garça - São Paulo, Brasil*
***Corresponding author**: fbio_moreno@yahoo.com.br
DOI: http://dx.doi.org/10.22456/2175-2745.107128 • **Received:** 31/08/2020 • **Accepted:** 05/06/2021

1. Introdução

Léxico é um conjunto de informações valiosas e são importantes para qualquer área de conhecimento, principalmente para as áreas de Letras, Linguística e Processamento de Linguagem Natural (PLN) [1]. Na sua aplicação, as possibilidades são infinitas no aprendizado, sendo exemplos a leitura, escrita, compreensão e tradução [2].

Léxicos são bases de grande volume de dados que têm em seu conjunto vários atributos linguísticos (etimologia, pronúncia, morfologia, sintaxe e definições) para cada um dos seus itens lexicais que podem servir a uma aplicação ou apenas como centralização e organização das informações [3].

Porém, segundo Paim [4] e Souza [5], isso é uma realidade distante no Brasil porque ferramentas e métodos de consultas para léxicos digitais são bem escassos para a Língua Portuguesa. Para Greghi [6], a criação de aplicações desse tipo é uma tarefa árdua e demorada que poderá ficar simplificada com a centralização dos dados em repositório que armazene todas as informações lexicais. Outro problema destacado é a maneira em que essas informações são organizadas e o tratamento dado a essas bases em suas construções que poderão acarretar em baixos desempenhos nas buscas. Lisbôa and Barbosa [7] também destacam que se uma palavra não for encontrada no dicionário, isso pode afetar todo processamento.

Um método para o desenvolvimento de ferramentas lexicais são as tabelas *hash* que necessariamente têm as informações centralizadas nessa estrutura de dados sendo eficientes para acessar grande volume de informações de modo rápido e otimizado, com potencial para resolver problemas de buscas

de multipalavras [8]. Exemplos de ferramentas com tabelas *hash* tem-se o léxico Urdu [9], o qual foi feito para uma língua indo-europeia amplamente falada no Paquistão e na Índia que se mostrou eficiente tanto para o tempo de pesquisa como nos requisitos de espaço e armazenamento, e o sistema *Morphological Analyzer using Hash Table*, ou apenas MAHT, o qual é um analisador morfológico para a Língua Espanhola que possui uma base de conhecimento lexical de quase cinco milhões de palavras, conseguindo atingir uma velocidade média de processamento de mais de 275 mil palavras por segundo, exatamente por usar tabelas *hash* na memória principal.

Já para a Língua Portuguesa são os trabalhos de léxicos que se utilizam de tabelas *hash*. Como exemplo temos o chatbot TICAL (acrônimo para *Tecnologia de Interação Conversacional sobre Assuntos Linguísticos*) [10] [11] que tem como base o Atlas Linguístico do Brasil (ALiB), que é o mais completo trabalho de pesquisa realizado na área de Geossociolinguística do Brasil até a atualidade [12] [13] e o LHiSPAR (Léxico Histórico do Paraná) [14] [15].

Esses sistemas apontam a eficácia e usabilidade da estrutura de dados *hash* trabalhada com léxicos. São, portanto, uma alternativa para desenvolvimento de ferramentas lexicais por serem fáceis de implementar [16] e também rápida por não ter a necessidade de análise e estruturação de banco de dados relacional, como é o caso do *chatbot* TICAL [10] [11].

Dessa forma, o objetivo geral deste trabalho é mostrar o desenvolvimento de uma ferramenta que empregue tabela esparsa a um léxico, gerando resultados de desempenho das funções *hash* implementadas, tendo como finalidade demonstrar efetividade de algumas dessas funções e suas eficácias no espalhamento, bem como determinar a melhor função. Cada dicionário tende a ter um espalhamento diferente e por essa razão dá-se a importância de conhecer a base e escolher o melhor algoritmo de espalhamento, para alcançar os melhores tempos de respostas. A escolha desse tema é relevante para demonstrar a eficiência da estrutura de dados na busca de respostas a questões inseridas por um usuário nessa tabela, usando os conceitos de PLN que são utilizados em uma Interface em Linguagem Natural.

Este trabalho está organizado da seguinte maneira: a segunda seção apresenta o aporte teórico envolvido no processo de desenvolvimento desta pesquisa; a terceira seção descreve os procedimentos metodológicos utilizados, contextualiza a temática, bem como os objetivos delineados neste trabalho; a quarta seção refere-se à ferramenta proposta para um Léxico das Ervas, chamado, Visual TaHs, suas funcionalidades e também exibe quatorze funções *hash* utilizadas em tal ferramenta; a quinta seção demonstra o comportamento, espalhamento, os tempos de busca e de *hash* com diversos tamanhos para tabela e quantidades de ervas; finalmente a sexta seção apresenta as considerações finais deste trabalho e sugestões para trabalhos futuros.

2. Aporte Teórico

Os léxicos são essenciais para o desempenho de qualquer aplicação em PLN [7] [17] que devem ter um tratamento adequado e um bom desempenho nas consultas [6]. Justamente o tempo de acesso ao léxico é motivo de pesquisa para vários autores, como Zock, Schwab and Rakotonanahary [18] que constataram que para uma busca eficaz é necessário um bom algoritmo, porém quando existem vários algoritmos, esses podem afetar a eficiência. Já Cercone, Krause and Boates [19] em suas pesquisas utilizaram tabelas *hash* para realizar o armazenamento e a busca de léxicos, utilizando-se do *hash* mínimo perfeito de Cichelli [20]. Specia e Rino [21] ressaltam que as tabelas *hash* são eficientes para acessar léxicos grandes de modo rápido e otimizado.

As tabelas *hash* são tipicamente usadas para indexação de grandes volumes de informação (como por exemplo, bases de dados). A implementação típica busca uma função *hash* que seja rápida para encontrar as respostas, não importando o número de registros na tabela e desconsiderando colisões. O ganho em relação a outras estruturas associativas (como um vetor simples) passa a ser maior conforme a quantidade de dados aumenta [22] [16].

2.1 Tabela Hash

Tabelas *Hash* são um tipo de estruturação para o armazenamento de informação de forma extremamente simples, fácil de implementar e intuitiva quando se trata de organizar grandes quantidades de dados. O conceito central é a divisão de um universo de dados a ser organizado em subconjuntos mais facilmente gerenciáveis. A estruturação da informação em tabelas *hash* visa principalmente permitir armazenar e procurar rapidamente grande quantidade de dados [16]. Essa estrutura tem conceitos iguais para um léxico em Língua Natural. Enquanto a tabela de símbolos para os Compiladores é necessária para catalogar as informações de identificadores (variáveis, funções, tipos de dados, etc.) e tem atributos que nos diz, por exemplo, qual seu nome, tipo ou escopo, já em um léxico para Língua Natural é necessário catalogar as palavras em sua categoria morfológica como substantivos, adjetivos, verbos, etc. e inserir em uma estrutura de dados que nos retorne o mais rápido possível essa palavra quando necessária e em que contexto está sendo empregada. A função *hash* pode ser um diferencial no Processador de Linguagem Natural de qualquer interface em língua natural.

A referida estrutura de dados é também conhecida por outras denominações como *tabela esparsa, tabela de dispersão, tabela de espalhamento* ou simplesmente *tabela hashing* e possui duas formas de ser implementadas [16]: tabelas com Endereçamento Direto e tabelas *Hash* Encadeada. Nos dois modos de implementação, as funções *hash* são as responsáveis por realizar a distribuição dos elementos na tabela e devem possuir técnicas de refinamento e de transformação da chave para endereço direto. Ou seja, uma vez fornecida a chave há uma busca única de chaves em uma tabela estática [16].

Ziviani [23] e Cormen et al. [16], assim como Knuth [24]:

definem alguns critérios para uma boa função *hash*:

1. Endereço *hash* facilmente calculado;
2. Fator de carga da tabela *hash* é elevado para um dado conjunto de chaves;
3. Os endereços de *hash* de um determinado conjunto de chaves são distribuídos uniformemente na tabela *hash* e uma função *hash* perfeita se diz ótima quando existe distribuição uniforme de endereços da tabela *hash*.

Conforme recomendações de Carreras-Riudavets et al. [25] e El-Abbadi, Khdhair and Al-Nasrawi [26], a tabela *hash* encadeada foi a escolhida para implementação da ferramenta Visual TaHs e a razão disso se justifica, uma vez que sua implementação não precisa limitar a quantidade de elementos e tampouco de recálculos de *hash*.

2.2 Tabela Hash e Léxico

Algumas aplicações correlatas foram analisadas para este trabalho. A primeira foi o sistema MAHT [25] que possui uma base de conhecimento lexical da Língua Espanhola de quase cinco milhões de palavras cadastradas em um banco de dados Oracle. Essas informações são carregadas para as duas tabelas *hash* inserindo sua chave, formas canônicas e formas gramaticais. Entretanto, não foram inseridos alguns tipos de pronomes, pela razão dos autores considerarem que o banco de dados iria aumentar consideravelmente e os tempos de pesquisa seriam altos.

Para Carreras-Riudavets et al. [25], o desempenho do sistema depende da função *hash* utilizada, sendo que essa, quando ruim, pode acarretar maus resultados. No projeto MAHT foram testadas várias funções e métodos, e foi desenvolvida uma função *hash* que apresentou ótimos resultados, com menos de duas comparações para localizar qualquer palavra.

No Léxico Urdu [9] seis tipos de funções *hash* foram testadas para verificar quais delas trabalhavam melhor com esse léxico. Foram utilizadas quatro bases Unicode para testar essas funções, sendo duas listas com palavras em inglês e duas listas com palavras em Urdu. Essas bases eram arquivos de texto. Com os resultados foi possível inferir as funções que obtiveram melhores e piores tempos de busca para esse léxico.

O chatbot TICAL [10] [15], o qual utiliza de linguagem natural, tem em sua base dois Léxicos: o ALiB e o LHiSPAR. Funciona com duas tabelas *hash*, uma para sinônimos das perguntas e outra com o léxico que inclui sinônimos também. A função *hash* utilizada para esse experimento foi a Mix 32 bits [27] e as informações estavam gravadas em dois arquivos XML. A chave utilizada no TICAL consiste de elementos ordenados e em conjunto com a tabela de sinônimo, a entrada para buscas pode ser feita de várias maneiras distintas. Para o cadastro das chaves (perguntas) são mantidas as palavras em evidências para um cálculo *hash* que identificará a posição na tabela e consequentemente o retorno da consulta.

El-Abbadi, Khdhair and Al-Nasrawi [26] desenvolveram uma função *hash* com bom desempenho para o Léxico da Língua Árabe, pois salientam que uma função *hash* selecionada ao acaso pode prejudicar o resultado gerando tempos lentos e, ainda, testar várias funções *hash* demanda muito tempo. Enfim, nessas aplicações os referidos autores indicam como uma ótima solução para Léxicos o uso de tabelas *hash*. Também perceberam a dificuldade em encontrar ou desenvolver uma função *hash* com bom desempenho, pois demanda tempo para implementação, comparação e principalmente para a realização de testes. Isso é um problema que dificulta a utilização dessa estrutura com os léxicos.

Porém, a usabilidade das tabelas *hash* e do léxico ALiB demonstraram ótimos resultados de tempo e de respostas compatíveis com o que era perguntado [28] [11]. Foram realizadas 734 perguntas, das quais 627 tiveram respostas compatíveis, sendo que das 107 perguntas que não tiveram êxitos, apenas 43 não tinham respostas e o restante foi por terem erros de português e ser fora do contexto.

Em sua nova fase, TICAL [11] conta com a base do projeto LHIsPAR com 4.171 verbetes e 349.656 palavras de documentos escritos durante os séculos de XVII, XVIII e metade do XIX pertencentes às antigas vilas e que hoje são municípios do estado do Paraná, de documentos da Casa da Memória de Curitiba e também do Arquivo Público do Estado do Paraná [14]. Por meio desse chatbot foi possível constatar a eficácia da utilização de léxicos com tabela *hash*. No entanto, a dificuldade de encontrar a função ideal a ser utilizada deu motivação para criação da ferramenta Visual TaHs que poderá dar parâmetros de tamanhos e de configurações para buscas eficazes.

3. Procedimentos Metodológicos

Este trabalho é uma pesquisa experimental que consiste essencialmente em determinar um objeto de estudo, selecionar as variáveis capazes de influenciá-lo e definir formas de controle e observação dos efeitos que a variável produz no objeto [29]. Para sua elaboração diversas etapas foram definidas para que fosse atingido seu objetivo geral que é desenvolver uma ferramenta que empregue tabela esparsa a um léxico, gerando resultados de desempenho das funções *hash* implementadas, tendo como finalidade determinar a melhor função. Para alcançar o objetivo deste trabalho, alguns procedimentos foram estruturados e definidos, tais como:

- Um levantamento teórico sobre funções *hash* por meio de livros das literaturas da área e de sítios de especialistas no assunto, como Cormen et al. [16], Aho, Sethi and Ullman [30] e Jenkins [27] foi realizado. A partir disso, decidiu-se pelo uso de tabelas encadeadas para a estrutura do léxico e essa escolha se deu pelo fato dessa estrutura permitir trabalhar com uma grande quantidade de dados;
- Para a catalogação das propriedades do léxico para a ferramenta proposta foram definidos como tópicos

para a estrutura de busca, aqueles que constam no livro "Léxico das Ervas" (nome original em alemão é *Lexikon der Kräuter*) [31]. Tal obra traz a história das ervas e descreve seus primeiros usos, bem como explica a maneira correta de secagem e o modo como guardá-las. Ao todo são descritas 105 espécies dessas. A escolha desse material para o experimento se justifica pela maneira organizada do seu léxico, definindo até doze características para cada uma das ervas. Isso já não ocorreu no chatbot TICAL [11] que utiliza um léxico com apenas dois atributos pergunta-resposta ou chave-significado;

- Para a estrutura do Léxico na ferramenta foram catalogadas todas as espécies contidas no livro mencionado das ervas, onde cada uma delas tinha 12 atributos: Nome Científico, Família, Outras Denominações, Sinônimos, Origem, História, Floração, Características, Habitat, Propriedades, Cozinha, Saúde e Cosmética. A Imagem da Erva também foi exibida. Todos os atributos das ervas descritos na Tabela 1 (veja exemplo para a erva AGASTACHE FOENICULUM) foram cadastrados em um Banco de Dados MySQL e depois exportada para um arquivo XML. A escolha dessa linguagem se justifica pela facilidade e agilidade de sua manipulação e não tem a necessidade de nenhum tipo de configuração, diferentemente dos Sistemas Gerenciadores de Banco de Dados (SGBDs);

- Para o desenvolvimento da ferramenta, a qual foi denominada como "Visual TaHs" e para a realização dos experimentos para comparação e análise dos desempenhos das funçõe *hash*, algumas etapas foram seguidas, tais como:

    1. Em sua primeira versão, a ferramenta foi desenvolvida em C++, porém algumas implementações e compatibilidades com outros componentes eram mais difíceis de serem feitas nessa plataforma. Por esse motivo foi reescrita em C# para viabilizar inúmeras possibilidades para outras aplicações [11];
    2. Para essa versão foram implementadas 14 funções *hash*, das quais 7 são encontradas em literaturas de disciplinas de Estrutura de Dados e de Compiladores, porém outras 7 funções foram projetadas por Robert Jenkins [32] [33], que é um pesquisador da computação e autor de várias funções *hash*. Essas são: Mix 32 bits [27], Mix 64 bits [27], Lookup3 32 bits [34], Lookup3 64 bits [34], CRC [35], Genérico CRC [36], One at a Time [27], Zobrist [37], Aho [30], PJW [38], Universal [16], Divisão [16], Multiplicação [16] e Dobra [39]. Para elaboração da tabela *hash* na linguagem C# foram utilizados os conceitos de orientação a objetos e listas para simular o mesmo processo na linguagem C++ quando se usa esse tipo de estrutura. O objeto "ervas" possui atributos que representam todos os campos do léxico das ervas. Todos os atributos foram criados com o tipo *string*, sendo desconsiderado o item imagem. Especificamente sobre as imagens foram salvas em uma pasta à parte e somente referenciados seus caminhos na tabela para que possam ser exibidas de forma mais rápida durante a consulta. Também é importante destacar que algumas ervas não possuíam certos atributos como a história, o sinônimo e a saúde, situação em que os respectivos campos ficam em branco quando é feita a busca. Para a chave de entrada na tabela *Hash* foram utilizados os nomes científicos das ervas por ser tratar de um nome único e que não se repete em todas as espécies;

- Desenvolvida a ferramenta, os experimentos foram realizados de modo que cada função fosse executada separadamente com cada conjunto de ervas e tamanhos de tabelas. Essas funções realizavam o mesmo teste em um total de dez vezes em cada uma das funções implementadas e com essas amostras foram realizadas médias. Por se tratar de resultados em microssegundos, esses apresentavam algumas variações nas execuções. Essa ação permitiu que as funções não fossem prejudicadas por falta de coleta de dados. Segundo Sawaya [40], o tempo de execução de um programa é um contraponto à compilação. As execuções de um programa incluem carregar e ligar bibliotecas necessárias para executar o programa, otimização dinâmica do programa e execução desse de fato. Os resultados obtidos para a análise e comparação foram feitos por meio de relatórios gerados pela ferramenta Visual TaHs. O tempo para cada experimento variava de acordo com o tamanho do léxico e da tabela, sendo que todo o experimento levou cerca de três meses para ser concluído;

- Com relação aos tempos obtidos nos experimentos, os valores finais apresentados nos relatórios constavam a soma de todos os cálculos de *hash* e das buscas realizadas para cada uma das ervas do léxico. Os tempos estavam em milissegundos. Com a finalidade de calcular uma média desses parâmetros (cálculo de *hash* e busca) para as funções foi realizada a divisão do valor total pela quantidade dos elementos do léxico. O resultado obtido foi um valor fracionário pequeno para realizar comparações e, por essa razão, esse número foi convertido para microssegundos;

- Para a execução da ferramenta e análise do experimento, visando escolher a melhor função *hash* para utilizar com o Léxico das Ervas, testes de desempenho, espalhamento e os tempos de cálculos de *hash* e de busca foram realizados por meio de um notebook com o processador i7, 8 Gigabytes de memória RAM, 1 Terabyte de disco rígido e sistema operacional Windows 7. Para

Tabela 1. Estrutura Léxico das Ervas.

Nome Científico	AGASTACHE FOENICULUM
Família	Lamiáceas (Lamiaceae)
Outros Nomes	Hissopo-anisado
Sinônimos	Agastache anethiodora
Origem	América do Norte
História	Esta planta já era conhecida pelos indígenas norte americanos devido aos seus valores nutritivo e medicinal. Foi introduzida na Europa por apicultores, pois as suas flores constituem uma excelente fonte de alimento para as abelhas
Floração	De junho a setembro.
Características	Planta vivaz, resistente no Inverno, cujo caule pode atingir 90cm de altura tura. Possui grandes inflorescências de cor púrpura e folhas verde-prateadas, com comprimento até 8 cm, sabor adocicado e cheiro a anis
Habitat	Esta planta cresce bem em vasos ou no jardim, em sítios expostos ao sol ou em locais semi-sombrios, com solo permeável. Na Europa Central não necessita de ser protegida da geada.
Propriedades	Destacam-se efeitos anti-inflamatórios e digestivos. As folhas devem ser colhidas antes de a planta florir, e podem ser utilizadas frescas ou secas
Cozinha	As folhas frescas podem ser utilizadas na preparação de saladas e na produção de licores; as flores são um excelente elemento decorativo em saladas e sobremesas.
Saúde e Cosmética	A partir das folhas secas pode-se fazer um chá que alivia dores de garganta, constipações e que acalma o estômago. As flores, frescas ou secas, servem de decoração em arranjos florais.
Imagem	

cada um dos testes realizados, a máquina era reiniciada e limpado o cache por meio de utilitários do sistema operacional. Para todos os testes foram utilizadas as mesmas amostras do dicionário. Essa limpeza do cache e o procedimento de reiniciar o Sistema Operacional eram necessários, porque se realizássemos o mesmo teste em sequência, os valores obtidos seriam menores. O objetivo aqui era encontrar uma média com o computador iniciando sem nenhuma informação em suas memórias;

- Para visualização foram elaborados relatórios e resultados quantitativos (esses obtidos com vários testes que demonstram os tempos alcançados na busca e no cálculo do *hash*) que têm como finalidade analisar os espalhamentos, as colisões e os tempos executados pela ferramenta para cada função de dispersão implementada. Esses relatórios e resultados quantitativos foram feitos anterior ao experimento, pois foram necessários para a contextualização e visualização dos experimentos;

Concluídos os relatórios, os experimentos foram executados e realizados contendo o tempo do cálculo de *hash* e o tempo de busca foram exportados para o programa Microsoft Excel com o intuito de realizar médias e elaborar gráficos de comparações que serão vistos nos experimentos:

- **Primeiro experimento**: foram utilizados 1024 endereços para tabela *hash* e foram utilizadas as 105 ervas definidas em Kothe [31]. Esse primeiro experimento serviu para uma análise inicial das funções, sendo realizadas comparações de espalhamento e de tempo;

- **Segundo experimento**: foi utilizada a mesma quantidade de endereços para tabela do experimento anterior (1024 endereços) e para uma análise mais profunda foram realizadas dez sobrecargas de 1000 (mil) ervas, totalizando no final 10000 (dez mil) ervas. Essa sobrecarga consiste na geração aleatória de ervas que foram criadas automaticamente pela ferramenta, sendo que todas as funções utilizaram o mesmo conjunto de elementos. Com as amostras obtidas nesse experimento foi possível acompanhar o desempenho das funções *hash* durante o aumento dos dados, realizando assim análises da performance de cada função;

- **Terceiro experimento**: uma sobrecarga de 100.000 (cem mil) ervas foi aplicada em diferentes tamanhos de tabela *hash*, sendo o tamanho utilizado na base 2 ($2^8 - 2^{20}$). Esse experimento serviu para traçar um perfil mais detalhado em relação ao comportamento das funções entre os tamanhos do léxico e o tamanho da tabela. Como resultado, foi possível determinar padrões para cada função.

4. Ferramenta Desenvolvida

Esta seção visa apresentar a ferramenta proposta nesta pesquisa chamada de Visual TaHs. Esse sistema analisa as funções esparsas por meio de suas buscas, cálculos do número do *hash* e espalhamento.

A ferramenta foi desenvolvida na linguagem C# utilizando o Visual Studio 2010 e sua interface está ilustrada na Figura 1 que contém na parte superior seu Menu de Opções e na parte central os campos para exibição das informações da erva e no lado direito está o *grid* que contém todos os elementos da tabela *hash* para que as consultas possam ocorrer na tabela esparsa e os campos para exibição sejam preenchidos.

Para a chave de entrada na tabela *Hash* utilizou os nomes científicos das ervas por ser tratar de um nome único e que não se repete em todas as espécies. Exemplos: ACHILLEA MILLEFOLIUM, ALLIUM CEPA, ALLIUM URSINUM, VERBENA OFFICINALIS e CHAMAEMELUM NOBILE.

Na ferramenta Visual TaHs existe a possibilidade de realizar as operações de CRUD (*Creat, Read, Update and Delete*), operações típicas dos Bancos de Dados, e todas as informações das ervas podem ser visualizadas em tela e seus resultados de tempos, espalhamento e colisões podem ser analisados em relatórios. Ainda, possui as seguintes funcionalidades:

1. *Quantidade de chaves* – essa opção altera o número de endereços da tabela;
2. *Sobrecarga de elementos* – função que gera de forma aleatória as ervas e seus atributos, sendo que a quantidade de elementos a ser criada é informada pelo usuário do sistema;
3. *Salvar XML* – nessa opção é possível salvar todo o léxico com suas alterações para que possa ser acessado novamente em outra ocasião. A Figura 2 ilustra o caso de uso dessa ferramenta.

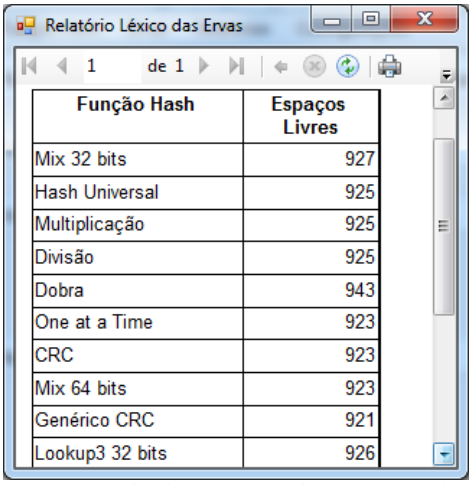
Para as ervas, uma classe foi criada contendo todos os seus atributos. A lista encadeada é representada por uma classe de mesmo nome que possui como atributo outra lista que contém os objetos ervas. Para cada uma das funções *hash* foi desenvolvida uma classe contendo as equações para o cálculo do *hash*. O diagrama de classe pode ser visto na Figura 3.

Visual TaHs gera relatórios e gráficos que são a peça fundamental na comparação de desempenho entre as funções *hash*. Para que essa análise fosse realizada, os seguintes relatórios foram elaborados: Colisões, Colisões Agrupado, Elementos Livres na Tabela *Hash*, Listagem *Hash* X Elementos e Desempenho.

O *Relatório de Colisões* possui dois gráficos que ilustram as colisões realizadas pela função *hash*. O gráfico de torta demonstra a quantidade de endereço utilizados da tabela *hash* (separado pela quantidade de colisões) e o gráfico de colunas ilustra o espalhamento do léxico realizado na tabela. Um exemplo desse relatório pode ser visto na Figura 4.

O *Relatório de Colisões Agrupado* ilustra o mapeamento realizado na tabela de duas maneiras: o primeiro gráfico faz o mapeamento dos elementos nos endereços, sendo possível analisar onde estão os picos de colisões; o segundo gráfico demonstra o mesmo mapeamento de forma ordenada pela quantidade de ervas por endereço, sendo assim, é possível visualizar se o espalhamento conseguiu realizar um bom trabalho. A Figura 5 mostra o exemplo de um relatório gerado.

No *Relatório de Elementos Livres na Tabela Hash* são fornecidas as quantidades de espaços livres na tabela após a aplicação do léxico. Esse valor é importante para a comparação e definição do melhor espalhamento como pode ser visto na Figura 6.



Função Hash	Espaços Livres
Mix 32 bits	927
Hash Universal	925
Multiplicação	925
Divisão	925
Dobra	943
One at a Time	923
CRC	923
Mix 64 bits	923
Genérico CRC	921
Lookup3 32 bits	926

Figura 6. Relatório de Espaços Livres na Tabela Hash.

O *Relatório Listagem Hash* X Elementos exibe a lista dos elementos do léxico e seu *hash* correspondente. Com seu resultado é possível identificar as ervas que sofreram colisões e as que estão alocadas sozinhas nos endereços da tabela. Na Figura 7 é mostrado um exemplo deste relatório.

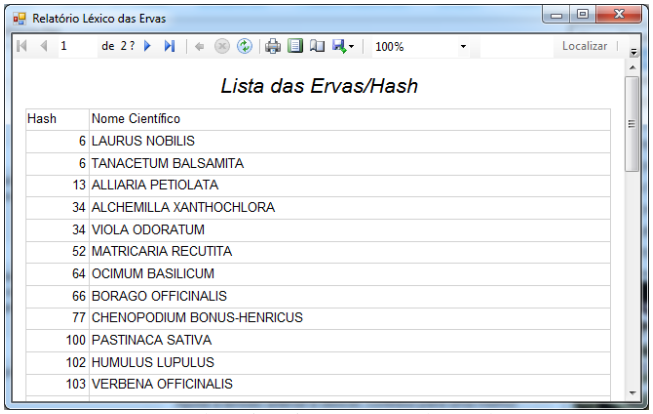


Figura 7. Relatório Listagem Hash X Elementos.

O *Relatório de Desempenho* é o relatório principal, sendo base para todos os experimentos. Pode ser realizado individualmente ou com todas as funções. Esse relatório fornece os tempos de cálculo de *hash*, busca e o tempo total, sendo esses contabilizados pela ferramenta que os disponibiliza em

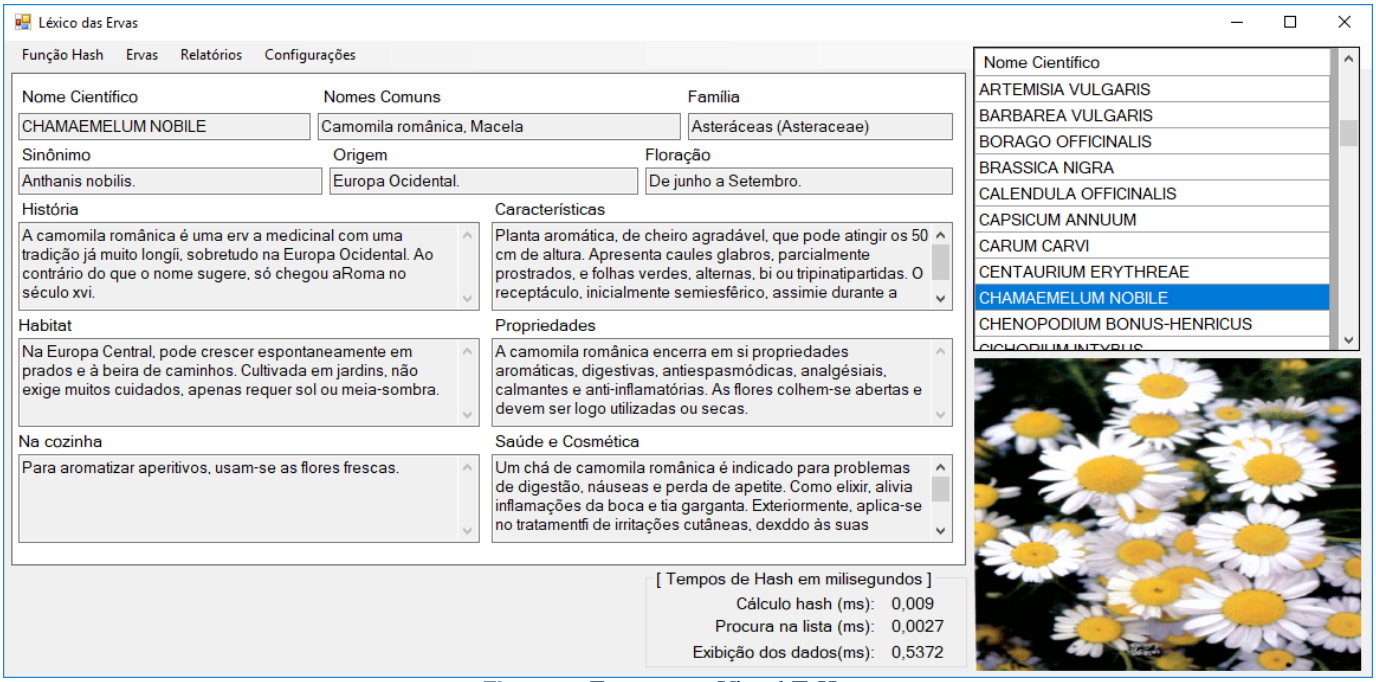
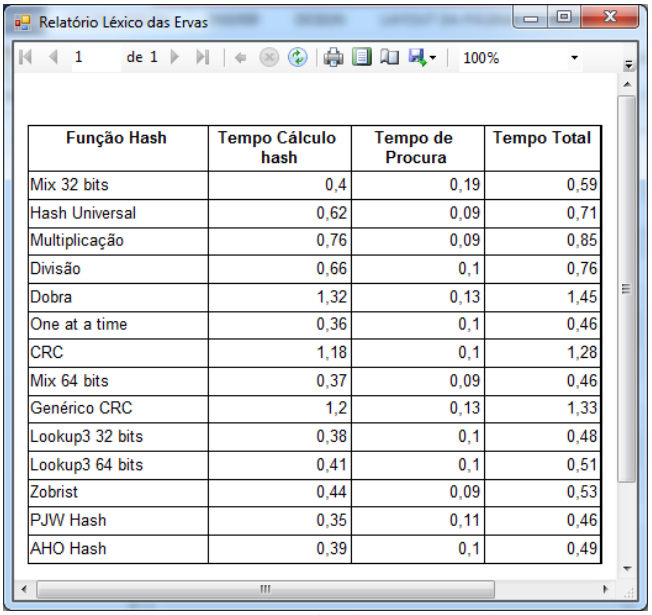


Figura 1. Ferramenta Visual TaHs.

milissegundos. Um exemplo deste relatório pode ser visto na Figura 8.



Função Hash	Tempo Cálculo hash	Tempo de Procura	Tempo Total
Mix 32 bits	0,4	0,19	0,59
Hash Universal	0,62	0,09	0,71
Multiplicação	0,76	0,09	0,85
Divisão	0,66	0,1	0,76
Dobra	1,32	0,13	1,45
One at a time	0,36	0,1	0,46
CRC	1,18	0,1	1,28
Mix 64 bits	0,37	0,09	0,46
Genérico CRC	1,2	0,13	1,33
Lookup3 32 bits	0,38	0,1	0,48
Lookup3 64 bits	0,41	0,1	0,51
Zobrist	0,44	0,09	0,53
PJW Hash	0,35	0,11	0,46
AHO Hash	0,39	0,1	0,49

Figura 8. Relatório de Desempenho.

Visual TaHs demonstra sua versatilidade em utilizar uma estrutura de dados para realizar as mesmas operações comuns de um banco de dados (incluir, editar, alterar e excluir) com o Léxico das Ervas. Além disso, a ferramenta é capaz de traçar uma visão de desempenho por meio de relatórios e gráficos demonstrando o espalhamento, o tempo de busca e o tempo de cálculo de *hash*. Esses parâmetros possibilitam ao operador da ferramenta escolher a melhor função *hash*.

Para a ferramenta Visual TaHs, além da catalogação do léxico, foram implementadas 14 funções *hash* descritas anteriormente.

5. Experimentos

Funções importantes para os testes foram disponibilizadas. Uma delas é a opção de alterar a quantidade de chaves da tabela, sendo que a configuração padrão está com 1024 chaves (0 ao 1023). Também foi implementada a opção de sobrecarga da função que consiste em uma função que irá inserir uma quantidade de ervas informada pelo usuário com palavras embaralhadas criadas aleatoriamente, sendo assim possível acompanhar o processo, o espalhamento e desempenho de cada função *Hash*.

Para isso, um experimento foi realizado e detalhado como demonstração do software Visual TaHs, sendo uma das maneiras de ser utilizada para auxiliar no aprendizado desse conceito. Nesse experimento são realizados testes nas quatorze funções. Essas avaliações são dos espalhamentos, dos tempos de cálculo do *Hash* e de busca dos elementos na tabela. Foi utilizado o tamanho padrão configurado pelo software de 1024 posições na tabela, sendo da posição 0 ao 1023. Os testes foram iniciados com 105 ervas e depois uma sobrecarga

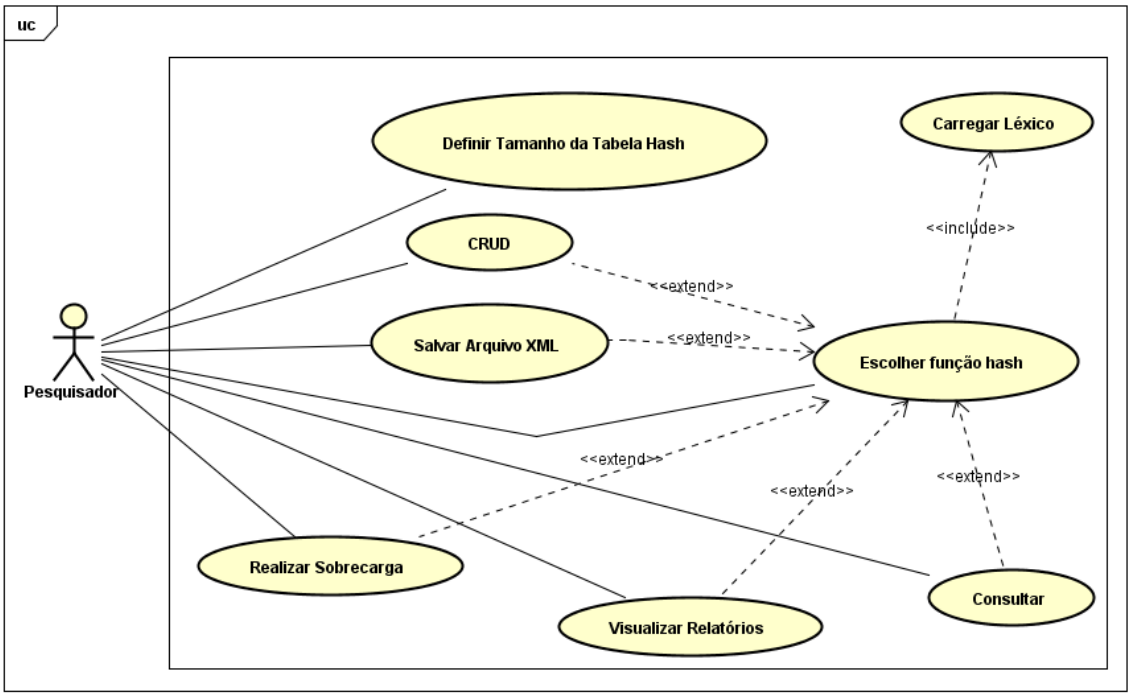


Figura 2. Caso de Uso Visual TaHs.

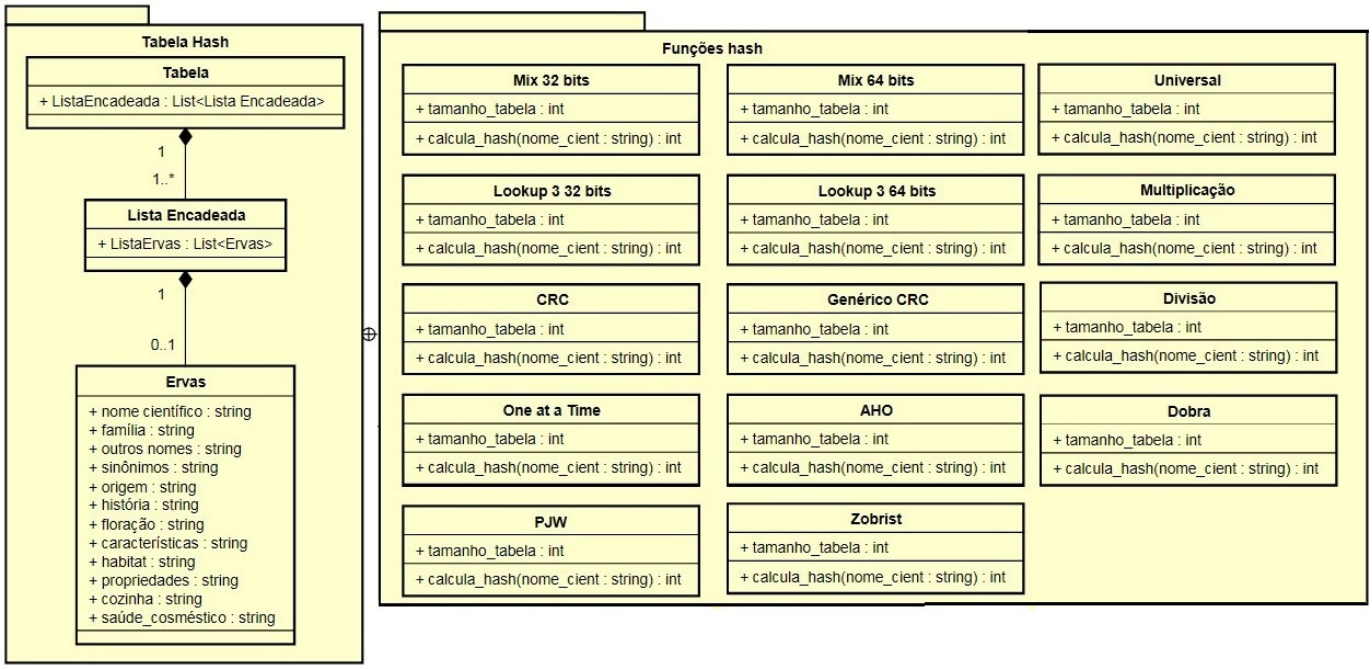


Figura 3. Diagrama de Classe – Tabela Hash.

foi efetivada usando uma proporção de 1000 elementos até contabilizar 10000 ervas, empregando sempre os mesmos elementos para todas as funções. As amostras desse experimento foram realizadas em torno de quatro horas, cujo tempo se deu em razão das coletas dos resultados e geração aleatória de dados (função do software que cria aleatoriamente dados para tabela sem ter a necessidade de digitação). Os resultados obtidos por meio da aplicação, ilustrados na Figura 9, demonstraram equilíbrio no quesito de ocupação dos espaços da tabela. A única função que não teve um bom aproveitamento de seus espaços é a função de Dobra que ao final do experimento ficou com 940 endereços livres na tabela.

Lembrando que essa função é necessária transformar o valor da chave (o nome cientifico da erva) em um valor numérico. Com esse valor, o método realiza uma dobra "como se fosse uma folha de papel", de maneira que os dígitos se

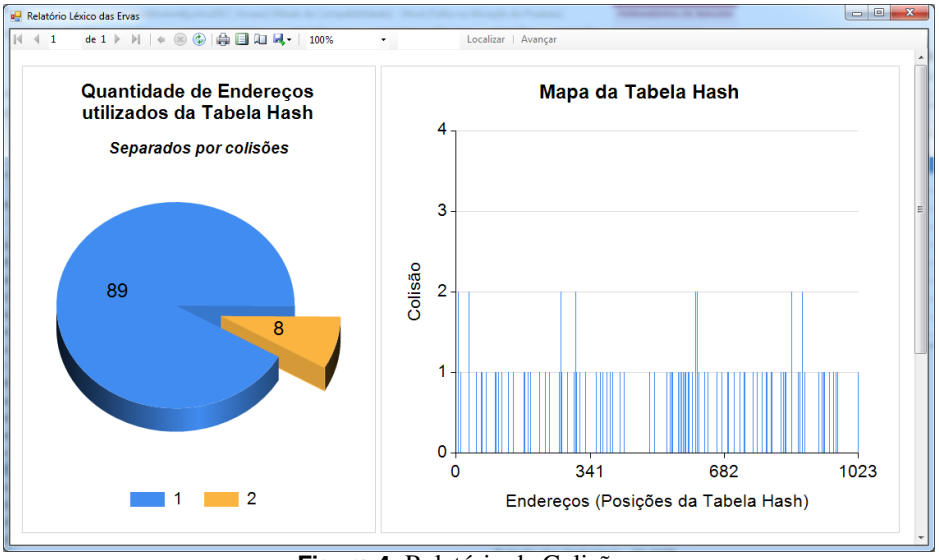


Figura 4. Relatório de Colisões.

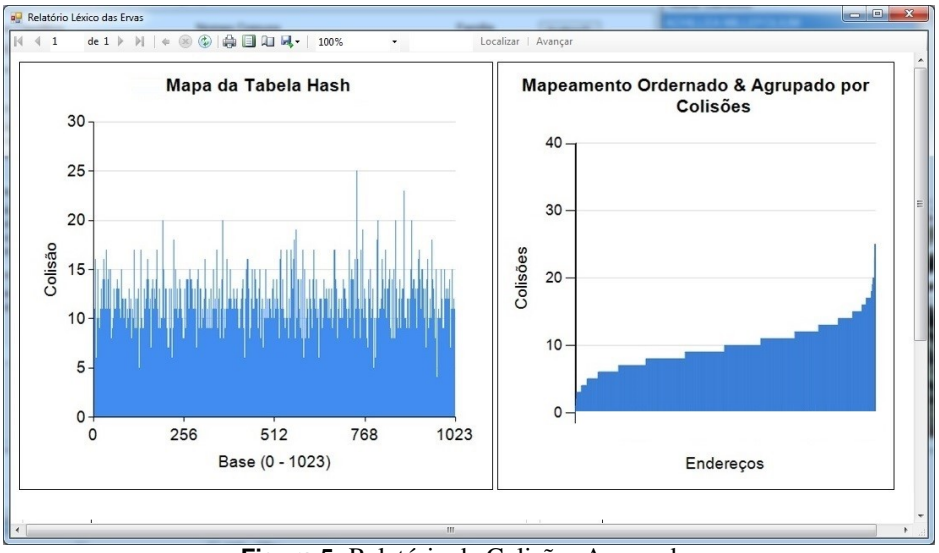


Figura 5. Relatório de Colisões Agrupado.

sobreponham. Esses valores são somados e não são levados em consideração o vai um [41]. A Figura 10 demonstra esse método:

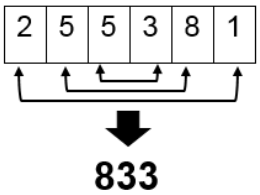


Figura 10. Função Dobra.

No exemplo da Figura 10, o valor 255381 realiza a dobra desses valores ficando 552 + 381, sendo o resultado 833. Essa operação é executada até que valor final seja menor que o tamanho da tabela. Outra maneira de se realizar esse conceito no procedimento de Dobra, os valores são convertidos para binário e, ao invés da soma, é utilizado o *ou exclusivo*. A Figura 11 ilustra esse processo.

Como o exemplo da Figura 11, o valor 71 realizou a dobra e a conversão para binário de seus valores, realizando a operação 0111 XOR 0001 gerou o resultado 0110 ou 5 em decimal.

E nessa última maneira, operações binárias foram implementadas e testadas nos experimentos. No entanto, algumas tabelas possuíam o valor mais alto que a chave e para uma melhor divisão na tabela dos elementos, a seguinte expressão foi utilizada para melhorar o espelhamento: chave=(chave*chave) elevado ao tamanho_tabela.

O valor da chave é multiplicado pelo próprio valor e depois executa uma operação de *ou exclusivo* com o tamanho da tabela. Caso o tamanho da chave obtido continue com o valor maior que a tabela, outro processo de Dobra ocorre.

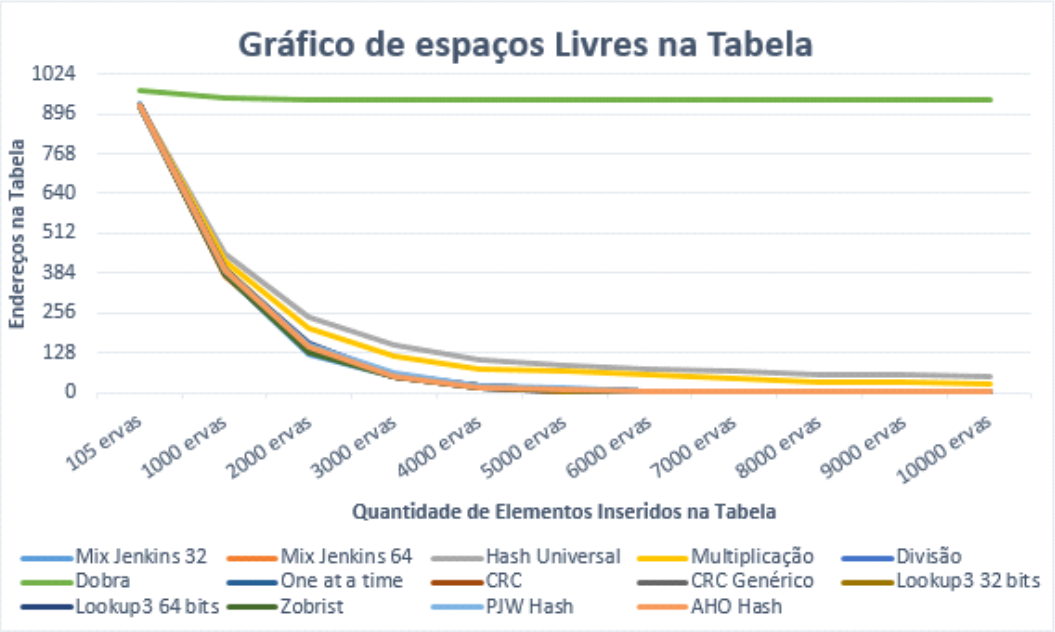


Figura 9. Gráfico dos Espaços Livres na Tabela.

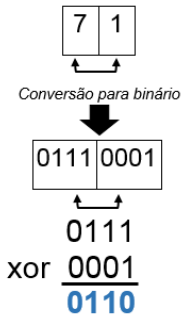


Figura 11. Dobra com *ou* exclusivo.

Toda a técnica da referida função foi implementada seguindo as instruções de Szwarcfiter and Markenzon [41]. Os ajustes implementados foram necessários para melhorar o espalhamento obtido anteriormente. O tempo dessa função obtido foi em decorrência das buscas, sendo que o algoritmo perdia muito tempo em realizar comparações das chaves na grande lista encadeada.

Os tempos de cálculos do número *hash* de todas as funções foram em milissegundos e não apresentaram o mesmo comportamento que foi mostrado anteriormente. Como demonstrado na Figura 12, Dobra, apesar de não conseguir utilizar os endereços adequadamente (Figura 9), realizou mais rápido o cálculo do *hash* do que as funções *One at a Time*, CRC e CRC Genérico que aproveitaram melhor os espaços da tabela (Figura 9). Já na Figura 12 é possível notar que sete funções foram as melhores e não modificaram seu tempo com o aumento dos elementos. Essas funções foram a Mix Jenkins 32 bits, Mix Jenkins 64 bits, Lookup 32 bits, Lookup 64 bits, Zobrist, PJW e a função Aho.

Os tempos de busca pelos elementos na tabela não tiveram influência no tempo final e, portanto, não alteraram a disposição do gráfico apresentado na Figura 13. Por meio do Visual TaHs também é possível visualizar a disposição (espalhamento) dos elementos da tabela, podendo isso, ser contemplado no momento do entendimento do conteúdo em sala de aula em uma aplicação, por exemplo, na disciplina de Estrutura de Dados. Na Figura 13 está ilustrada a função Mix 32 bits (uma das melhores funções), Dobra (a função com pior espalhamento) e CRC Genérico (a função com o pior tempo).

Com esse experimento foi possível identificar quais funções são mais eficientes para o Léxico das Ervas [31] e também conferir os espalhamentos de cada função, sendo que esses não são um fator determinante para o tempo final, diferente do cálculo do *hash* que é a principal causa para uma busca rápida ou mais lenta.

Com os resultados de todas as funções foi possível fazer um comparativo entre os melhores tempos alcançados, demonstrados na Figura 14. Entre parênteses estão os tamanhos da tabela que os valores foram obtidos.

Pela variação mínima de tempo, as oito primeiras funções da Figura 14 podem ser consideradas uma boa opção para o Léxico das Ervas. Os sete melhores resultados dentre essas oito funções de melhores tempos utilizaram o tamanho da tabela entre 524288 e 1048576. Em contrapartida, o pior resultado foi do *hash* Dobra [39].

Com o experimento foi possível traçar características para cada uma das funções em relação ao tamanho da tabela e também de desempenho, chegando assim as seguintes conclusões:

- Mix 32 bits [27], Mix 64 bits [27], One at a Time [27] , PJW [38], AHO [30], Lookup3 64 bits [34], Lookup3 32 bits [34] e Zobrist [37] foram as funções que obtive-

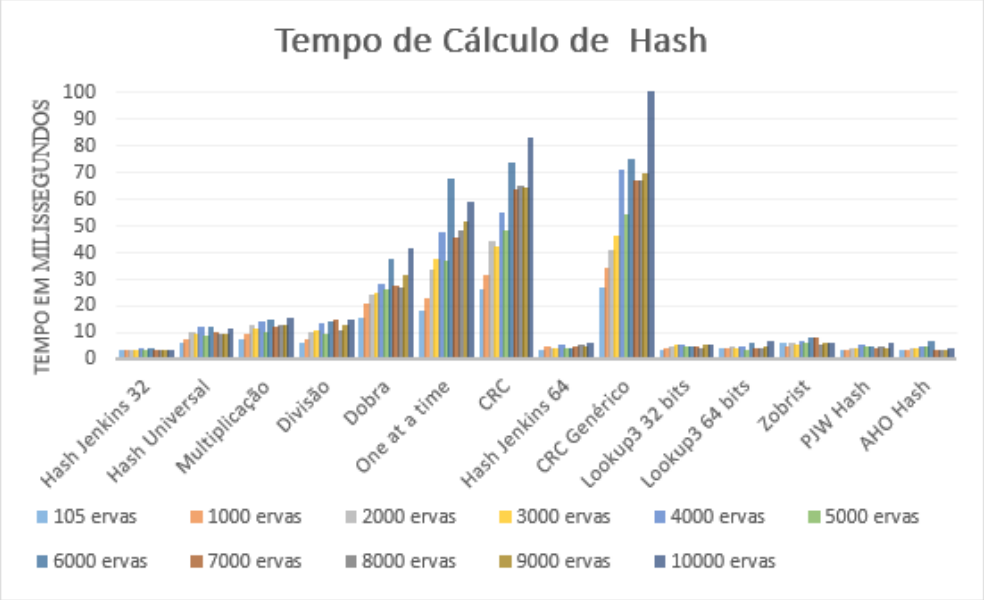


Figura 12. Gráfico Tempo de Cálculo Hash.

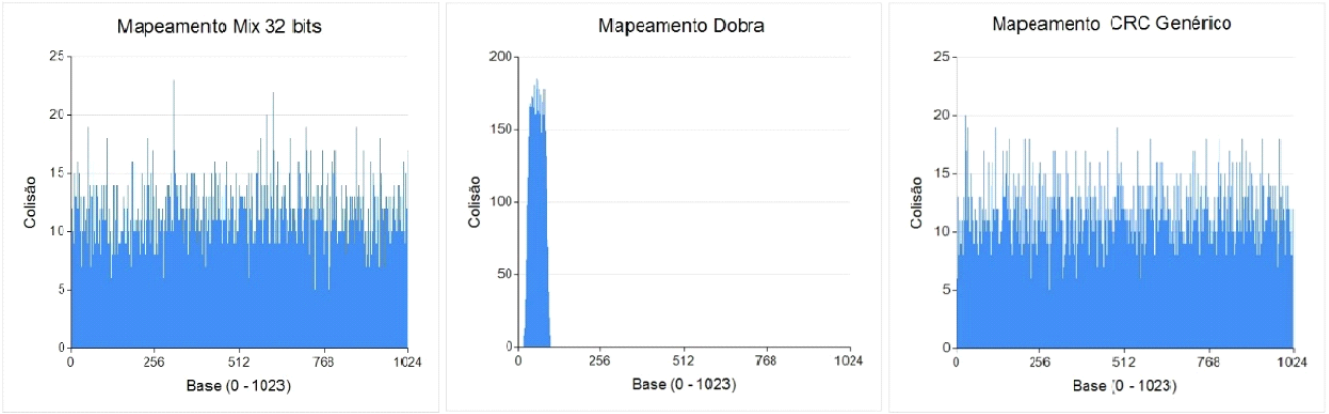


Figura 13. Gráfico de Espalhamento.

ram os melhores resultados com o Léxico das Ervas e trabalharam melhor com tamanhos de tabelas partindo de 4% da dimensão do léxico;

- CRC [35] e Genérico CRC [36] que possuem os mesmos algoritmos, contudo, valores diferentes para o *hash*, apresentaram o mesmo tempo, os quais são melhores com tabelas partindo de 16% da quantidade de elementos do léxico;

- Universal, Multiplicação e Divisão [16] apresentaram variações para o cálculo do *hash*. No geral, tiveram um melhor aproveitamento quando a tabela tinha a partir de 2% em relação à quantidade de elementos do léxico;

- Dobra [39] apresentou um desempenho constante, porém com o pior tempo apresentado. Os seus melhores tempos ocorreram com o tamanho equivalente a 66% em consideração ao volume do léxico;

- Para os testes realizados por Madeira [42], as tabelas perdiam sua eficácia ao alcançar 50% de sua utilização quando empregados para processamento gráfico. Com o léxico, as funções tiveram comportamentos diferentes. Em todos os casos o *hash* foi melhor tendo sobrecargas em tabelas com tamanhos menores do que a quantidade de registros.

Com os resultados e análises obtidas com a ferramenta "Visual TaHs" é possível inferir que as funções *hash* Mix 32 bits [27], Mix 64 bits [27] e One at a Time [27], PJW [38], AHO [30], Lookup2 64 bits [34], Lookup 32 bits [34] e Zobrist [37] são as melhores alternativas para serem utilizadas com léxicos, tanto pelo espalhamento quanto pelo tempo que demonstraram, e ainda, que essas funções têm um melhor desempenho quando existe uma grande quantidade de elementos no léxico.

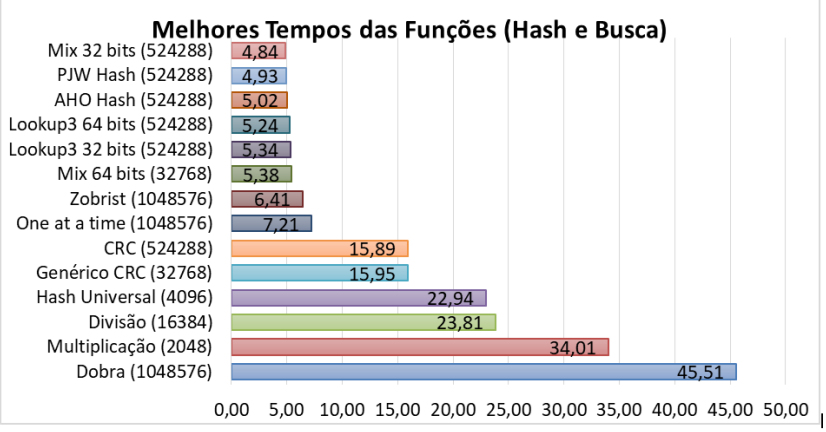


Figura 14. Gráfico de Espalhamento.

6. Considerações Finais

Por meio da ferramenta Visual TaHs desenvolvida neste trabalho foi possível gerar resultados de desempenho das funções *hash* implementadas, determinar as melhores funções, bem como comprovar sua eficácia nos tempos apresentados atingindo o objetivo desta pesquisa. O desenvolvimento de tal software foi possível por meio de um aporte teórico que permitiu uma investigação sobre algumas ferramentas que utilizam léxicos e suas experiências com a estrutura de dados do tipo *hash*. Este trabalho também forneceu embasamento sobre as funções esparsas dando condições para escolha do tipo de tabela a ser utilizada.

O dicionário escolhido para ser o experimento desta ferramenta foi o "Léxico das Ervas", o qual foi utilizado por ter uma estrutura bem definida, sendo suas características distribuídas por 12 atributos na interface do Visual TaHs.

Com a ferramenta desenvolvida e o léxico catalogado, três experimentos foram realizados:

1. No primeiro experimento foram analisadas as funções esparsas com as 105 ervas do Léxico e os resultados obtidos mostraram o comportamento das funções nos espalhamentos e nos tempos, porém a quantidade pequena de elementos do léxico resultou em valores semelhantes das funções;

2. No segundo experimento foi realizada uma sobrecarga de 1000 (mil) até o total de 10000 (dez mil) ervas e com esses testes foi possível analisar o comportamento de espalhamento das funções e o tempo de cada uma delas, sendo que, o mau espalhamento acarretou em tempos de buscas muito altos e o tempo de cálculo de *hash* pode ser um fator determinante para o tempo total. Funções com algoritmos simples (por exemplo, a função *hash* Divisão [16]) não são sinônimos de desempenho com relação ao tempo;

3. No terceiro experimento foi realizada uma sobrecarga de 100000 (cem mil) ervas em diferentes tamanhos de tabela. Uma característica encontrada com esse experimento foi que funções *hash* conseguem trabalhar bem com léxicos e ter bons rendimentos com tamanhos menores do que a quantidade de elementos nessa estrutura de dados. Com esse experimento foi possível definir padrões para cada uma das funções e também determinar as melhores.

O presente trabalho buscou refletir as interfaces dos estudos do léxico e os resultados demostrados nesses experimentos também servirão para encorajar a construção de léxicos digitais com tabelas *hash* que podem funcionar em vários meios como *desktop*, *Web* e *mobile*.

Além dos objetivos atingidos, pode-se destacar que a ferramenta construída poderá ser utilizada no âmbito da pesquisa para realizar novos experimentos com outros tipos de funções de dispersão e diferentes chaves do léxico, auxiliando assim outros pesquisadores a decidirem pela melhor função esparsa que possa ser utilizada.

Para a realização deste trabalho algumas limitações foram encontradas, sendo a primeira delas a realização de consultas com outras ferramentas ou aplicativos abertos, a qual ocorre perda de desempenho das funções, prejudicando assim os resultados. Por essa razão foi necessária a utilização de um computador dedicado para pesquisa. O tempo dos relatórios foi outra limitação encontrada, pois dependendo do tamanho do léxico, esse resultado podia demorar horas para ser finalizado.

Como trabalhos futuros almeja-se inserir a opção para realizar testes de desempenho em diferentes tipos de léxicos, como por exemplo, no Léxico das Orquídeas [7]. Ainda, pretende-se tornar disponível o código fonte para outros pesquisadores poderem entender o conceito de *hash* implementado, bem como ser um software de apoio ao ensino de tabelas *hashing* na disciplina de Estrutura de Dados para o curso de Ciência da Computação. Um levantamento realizado por Barbosa and Parreira Júnior [43] ressalta o fato de existirem poucas ferramentas educacionais que versam sobre Tabela *Hash*.

Contribuição dos Autores

Todos autores contribuíram igualmente para este trabalho.

Referências

[1] BARROS, L. A.; ISQUERDO, A. N. **O léxico em foco**. São Paulo: Cultura Acadêmica, 2010.

[2] VASQUES, P.; AGUILERA, V. Uma análise semântica do léxico em documentos históricos do Paraná. In: **X Seminário de Pesquisa em Ciências Humanas**. Londrina: Universidade Estadual de Londrina, 2017. p. 477–491.

[3] GONZALEZ, M.; LIMA, V. L. S. Recuperação de informação e processamento da linguagem natural. In: **XXIII Congresso da Sociedade Brasileira de Computaçao**. São Paulo: SBC, 2003. p. 347–395.

[4] PAIM, A. M. **Inferência de personalidade a partir de textos em português brasileiro utilizando léxicos**. Dissertação (Mestrado em Ciência da Computação) — Programa de Pós-Graduação em Informática Pontifícia Universidade Católica do Paraná, Curitiba, 2016.

[5] SOUZA, E. N. P. **Classificação de relações semânticas abertas baseada em similaridade de estruturas gramaticais na Língua Portuguesa**. Dissertação (Mestrado Multi-institucional) — Pós-Graduação em Ciência da Computação. Universidade Federal da Bahia, Salvador, 2014.

[6] GREGHI, J. G. **Projeto e desenvolvimento de uma base de dados lexicais do português**. Dissertação (Mestrado) — Instituto de Ciências Matemática e de Computação. Universidade de São Paulo, São Carlos, 2002.

[7] LISBÔA, A. R.; BARBOSA, C. R. S. C. Lexicon of orchids. **Procedia-Social and Behavioral Sciences**, Elsevier, Amsterdã, v. 95, p. 81–88, 2013.

[8] VILLAVICENCIO, A.; CASELI, H. M.; MACHADO, A. Identification of multiword expressions in technical domains: Investigating statistical and alignment-based approaches. In: **VII Brazilian Symposium in Information and Human Language Technology**. São Paulo: USP, 2009. p. 27–35.

[9] RIZVI, S. J.; HUSSAIN, M.; QAISER, N. Comparison of hash table verses lexical transducer based implementations of Urdu lexicon. In: IEEE. **Student Conference on Engineering, Sciences and Technology**. Piscataway, 2004. p. 29–29. Disponível em: <https://dx.doi.org/10.1109/SCONES.2004.1564764>.

[10] MANFIO, E. R.; MORENO, F. C.; BARBOSA, C. R. S. C. Professor Tical e Alib: Interação humano computador em diferente campo. In: **XIX Conferência Internacional sobre Informática na Educação**. Fortaleza: Nuevas Ideas en Informática Educativa, 2014. p. 782–787.

[11] MORENO, F. et al. Tical: Chatbot sobre o Atlas Linguístico do Brasil no WhatsApp. In: **XXX Brazilian Symposium on Computers in Education**. Maceió: SBC, 2015. p. 279–288.

[12] CARDOSO, S. A. **Atlas linguístico do Brasil**. Londrina: SciELO-EDUEL, 2014. v. 1.

[13] CARDOSO, S. A. M. et al. **Atlas linguístico do Brasil**: Cartas linguísticas 1. Londrina: SciELO-EDUEL, 2014. v. 2.

[14] ALMEIDA, J. E. B. Para a história do português paranaense. **Revista da ABRALIN**, São Cristóvão, v. 12, n. 2, 2013.

[15] MANFIO, E. R.; MORENO, F. C.; BARBOSA, C. R. S. C. Tical: Um chatbot que versa sobre assuntos linguísticos. In: **XIV Encontro Linguística de Corpus**. São Leopoldo: UFRGS, 2017. p. 51–51.

[16] CORMEN, T. H. et al. **Algoritmos**: Teoria e prática. Rio de Janeiro: Campus, 2002. v. 2.

[17] MACHADO, A. et al. Personalitatem lexicon: Um léxico em português brasileiro para mineração de traços de personalidade em textos. In: **XXX Brazilian Symposium on Computers in Education**. Maceió: SBC, 2015. p. 1122–1126.

[18] ZOCK, M.; SCHWAB, D.; RAKOTONANAHARY, N. Lexical access, a search-problem. In: **II Workshop on Cognitive Aspects of the Lexicon**. Beijing, China: Colling Organizing Committee, 2010. p. 75–84.

[19] CERCONE, N.; KRAUSE, M.; BOATES, J. Minimal and almost minimal perfect hash function search with application to natural language lexicon design. **XXX Computers & Mathematics with Applications**, Elsevier, Amsterdã, v. 9, n. 1, p. 215–231, 1983.

[20] CICHELLI, R. J. Minimal perfect hash functions made simple. **Communications of the ACM**, ACM, New York, USA, v. 23, n. 1, p. 17–19, 1980.

[21] SPECIA, L.; RINO, L. H. M. O desenvolvimento de um léxico para a geração de estruturas conceituais unl. In: **Série de Relatórios Técnicos do NILC**. São Carlos: NILC-ICMC-USP, 2002. p. 25.

[22] BOTELHO, F. C. **Estudo comparativo do uso de hashing perfeito mínimo**. Dissertação (Ciência da Computação) — Programa de Pós Graduação em Ciência da Computação. Universidade Federal de Minas Gerais, Belo Horizonte, 2004.

[23] ZIVIANI, N. **Projetos de algoritmos com implementações em Pascal e C**. São Paulo: Cengage Learning, 2011.

[24] KNUTH, D. E. **The art of computer programming 3**: Sorting and searching. Massachusetts: Addison-Wesley, Reading, 1973.

[25] CARRERAS-RIUDAVETS, F. J. et al. A morphological analyzer using hash tables in main memory (MAHT) and a lexical knowledge base. In: GELBUKH, A. (Ed.). **Computational Linguistics and Intelligent Text Processing**. Berlin, Heidelberg: Springer, 2012. p. 80–91.

[26] EL-ABBADI, N.; KHDHAIR, A. N.; AL-NASRAWI, A. Build electronic arabic lexicon. **The International Arab Journal of Information Technology**, Zarqa, v. 8, p. 137–140, 2011.

[27] JENKINS, R. J. **New hash function for hash table lookup**. 2013. Acesso em: 30 de jun. de 2020. Disponível em: <http://burtleburtle.net/bob/hash/doobs.html>.

[28] MANFIO, E. R.; MORENO, F. C.; BARBOSA, C. R. S. C. Professor Tical: Robô de conversação sobre dialetologia e geossociolinguística. In: **III Congresso Internacional de Dialetologia e Sociolinguística–Variação, Atitudes Linguísticas e Ensino**. Londrina: UEL, 2014. p. 48–48.

[29] GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2010. v. 5.

[30] AHO, A. V.; SETHI, R.; ULLMAN, J. D. **Compiladores**: Princípios técnicas e ferramentas. Rio de Janeiro: LTC, 1995.

[31] KOTHE, H. W. **Léxico das Ervas**. 1. ed. Lisboa: Dinalivro, 2009.

[32] JENKINS, B. ALGORITHM ALLEY: What makes one hash function better than another? Bob knows the answer, and he has used his knowledge to design a new hash function that may be better than what you're using now. **Dr Dobb's Journal-Software Tools for the Professional Programmer**, Redwood City, CA, v. 22, n. 9, p. 107–110, 1997.

[33] JENKINS, R. J. **Hash functions for hash table lookup**: Burtleburtle. 1997. Acesso em: 30 de jun. de 2020. Disponível em: <http://burtleburtle.net/bob/hash/evahash.html#newhash>.

[34] JENKINS, R. J. **Lookup3**: Burtleburtle. 2006. Acesso em: 30 de jun. de 2020. Disponível em: <http://burtleburtle.net/bob/c/lookup3.c>.

[35] JENKINS, R. J. **Hash CRC**: Burtleburtle. 2006. Acesso em: 30 de jun. de 2020. Disponível em: <http://burtleburtle.net/bob/c/crc.c>.

[36] JENKINS, R. J. **Generic CRC**: Burtleburtle. 2006. Acesso em: 30 de jun. de 2020. Disponível em: <http://burtleburtle.net/bob/c/gencrc.c>.

[37] ZOBRIST, A. A new hashing method with application for game playing. **International Computer-Chess Association Journal**, Amsterdã, v. 13, n. 2, p. 69–73, 1990.

[38] FLECK, M. **Hash functions**. 2000. Acesso em: 30 de jun. de 2020. Disponível em: <https://www.cs.hmc.edu/~geoff/classes/hmc.cs070.200101/homework10/hashfuncs.html>.

[39] SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Database Systems Concepts**. Nova Iorque: McGraw-Hill, Inc., 2005.

[40] SAWAYA, M. R. **Dicionário de informática & Internet**. São Paulo: NBL, 2002.

[41] SZWARCFITER, J. L.; MARKENZON, L. **Estruturas de Dados e seus Algoritmos**. Rio de Janeiro: LTC, 2010.

[42] MADEIRA, D. **Uma estrutura baseada em hash table para buscas otimizadas em octree em GPU**. Dissertação (Computação) — Universidade Federal Fluminense, Rio de Janeiro, 2010.

[43] BARBOSA, W. A.; PARREIRA JÚNIOR, P. A. Um mapeamento sistemático sobre ferramentas de apoio ao ensino de algoritmo e estruturas de dados. In: **II Congresso Brasileiro de Informática na Educação**. Campinas: Unicamp, 2013. p. 406–415.