UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HÉLIO HENRIQUE SADAO FERREIRA TAMANAHA

DESENVOLVIMENTO DE UM SISTEMA DE AUXÍLIO DO ENSINO DE INTELIGÊNCIA ARTIFICIAL UTILIZANDO VISUALIZAÇÕES DE ALGORITMOS

BAURU
2017

HÉLIO HENRIQUE SADAO FERREIRA TAMANAHA

DESENVOLVIMENTO DE UM SISTEMA DE AUXÍLIO DO ENSINO DE INTELIGÊNCIA ARTIFICIAL UTILIZANDO VISUALIZAÇÕES DE ALGORITMOS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Profª. Drª. Simone das Graças Domingues Prado

BAURU
2017

Hélio Henrique Sadao Ferreira Tamanaha DESENVOLVIMENTO DE UM SISTEMA DE AUXÍLIO DO ENSINO DE INTELIGÊNCIA ARTIFICIAL UTILIZANDO VISUALIZAÇÕES DE ALGORITMOS/ Hélio Henrique Sadao Ferreira Tamanaha. – Bauru, 2017- 40 p. : il. (algumas color.) ; 30 cm.
Orientador: Profª. Drª. Simone das Graças Domingues Prado
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho" Faculdade de Ciências
Ciência da Computação, 2017.
1. Tags 2. Para 3. A 4. Ficha 5. Catalográfica

Hélio Henrique Sadao Ferreira Tamanaha

DESENVOLVIMENTO DE UM SISTEMA DE AUXÍLIO DO ENSINO DE INTELIGÊNCIA ARTIFICIAL UTILIZANDO VISUALIZAÇÕES DE ALGORITMOS

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Orientadora: Profª. Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Profª. Drª. Andréa Carla Gonçalves Vianna
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Profª. Drª. Márcia A. Zanoli Meira e Silva
Departamento de Computação
Faculdade de Ciências
UNESP - BAURU

Bauru, treze de dezembro de 2017.

Resumo

Este projeto consiste no desenvolvimento de uma ferramenta para o auxilio do ensino de inteligência artificial utilizando a visualização de algoritmos, onde foi estudada a história de desenvolvimento de sistemas voltados para o ensino de conceitos de computação utilizando visualizações e como desenvolver ferramentas de visualização eficazes para o ensino. A ferramenta desenvolvida contêm diversas visualizações de algoritmos relevantes no campo da inteligência artificial, bem com um sistema de perguntas e respostas referentes as visualizações implementadas, sendo que é oferecida a opção de realizar o cadastro no site para poder adicionar, remover ou editar as perguntas novas, de acordo com as preferências do professor.

Palavras-chave: Visualização de Algoritmos, Inteligência Artificial,Aplicação Web.

Abstract

This Project is about the development of an application for teaching Artificial Intelligence using algorithm visualizations, where it was studied the history of important systems of the field, and also what techniques and concepts needed to be used to develop a system that actually helps in the learning of its users. It was also implemented a system of questions and answers, that offers to the teachers that use this system the possibility of adding, editing and excluding questions about each and every visualization, offering the possibility of customization.

Keywords: algorithm animation, artificial intelligence, web application.

Lista de ilustrações

Lista de abreviaturas e siglas

VA	Visualização de Algoritmo
DOM	Document Object Model
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JSAV	Javascript Algorithm Visualization Library
EJS	Embedded JavaScript
NPM	node package manager
SVG	Scalable Vector Graphics

Sumário

1 Introdução

O ensino é uma importante parte da sociedade, e justamente por isso educadores constantemente procuram novas formas de ensino com o intuito de aprimorá-lo (BYRNE; CATRAMBONE; STASKO, 1999) , facilitando o aprendizado e o foco dos alunos. O aprendizado da ciência da computação abrange o estudo de diversos subcampos, sendo que alguns são notoriamente difíceis, como a própria programação (JENKINS, 2001), que apesar de ser um assunto fundamental para a computação em geral, estudantes possuem uma grande dificuldade em dominá-lo.

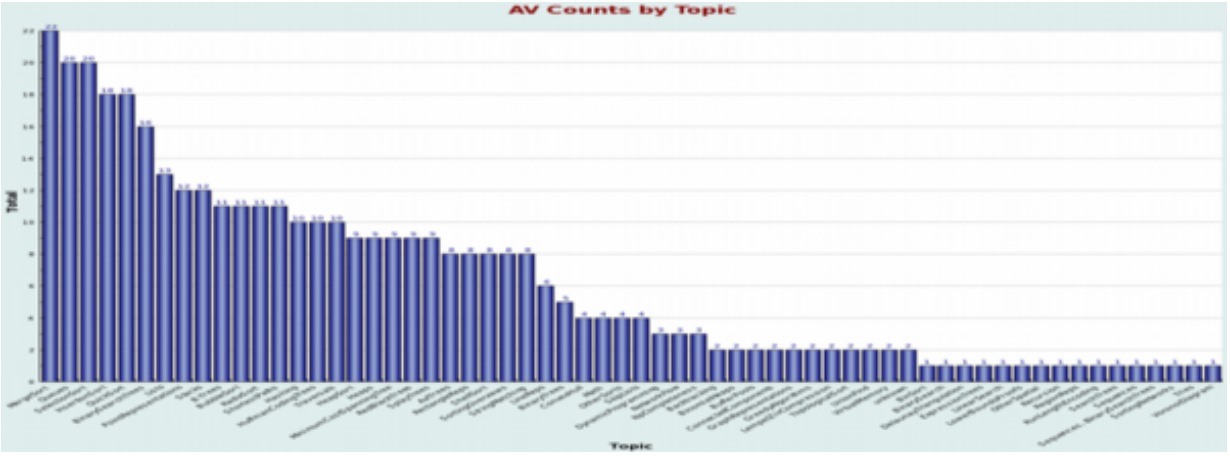
Com o intuito de auxiliar o ensino da programação, uma ferramenta intuitivamente procurada para ajudar a entender os algoritmos utilizados é a de visualização de algoritmos (KEHOE; STASKO; TAYLOR, 2001), que desde seu advento no final de 1970, evoluiu de softwares orientados a lotes que permitiam a criação de filmes animados para sistemas altamente interativos (HUNDHAUSEN et al. 2002). Apesar de haver discussões a respeito da eficácia em utilizar a visualização de algoritmos (VA) para a pedagogia (KEHOE; STASKO; TAYLOR, 2001), meta-estudos mais abrangentes apontam que existe um impacto significativamente positivo na utilização de animações, com a ressalva de que a simples visualização dos algoritmos não é suficiente para melhorar o aprendizado, sendo necessário a criação de ferramentas que permitem a maior interação dos alunos (HUNDHAUSEN; DOUGLAS; STASKO, 2002).

Apesar de seu efeito positivo no ensino de programação, a utilização de visualizações para compreender algoritmos não é amplamente utilizada em outras áreas da computação (SHAFFER et al., 2010), o que é exemplificado pela Figura 1, que mostra a contagem dos tópicos abordados em visualizações, sendo que pode-se observar que a grande maioria aborda o tema de ordenação e estrutura de dados.

Tendo em vista essa análise a respeito do atual conjunto de visualizações existentes, uma das áreas em que se pode observar a falta de ferramentas que auxiliem o seu aprendizado é a Inteligência Artificial, que atualmente possui as mais diversas aplicações, como o desenvolvimento de veículos capazes de navegarem pelo complexo tráfego de cidades sem o auxílio de um humano (ROSENZWEIG; BARTL, 2015).

Pelo elevado potencial de aplicação e complexidade da inteligência artificial, que utiliza conceitos de diversos outros campos como base, como a estatística, teoria de informação e a psicologia, para nomear alguns, fica claro que técnicas para auxiliar o ensino contribuíram para estimular a entrada e compreensão de seus conceitos por interessados na área, Tendo isso em vista, o objetivo desse trabalho é desenvolver uma aplicação que utiliza visualizações de algoritmos que aborde algoritmos utilizados pela inteligência artificial.

Figura 1 – Número de visualizações de algoritmos por tópicos



Fonte: Elaborada pelo autor.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Desenvolver uma plataforma visual para o auxilio do aprendizado de Inteligência Artificial.

1.1.2 OBJETIVOS ESPECÍFICOS

a) Definir os algoritmos que serão implementados e a forma com a qual as animações serão realizadas, com o enfoque em deixa-las adequadas para o ensino.

b) Análise de ferramentas de implementação: será analisada e definida qual ferramenta será utilizada para realizar a implementação de forma eficaz.

c) Definir a estrutura do projeto.

d) Implementar computacionalmente a ferramenta proposta.

e) Análise e Testes: Nesta etapa a ferramenta criada será testada e avaliada de acordo com a qualidade das representações realizadas.

2 Fundamentação Teórica

2.1 Definição de Visualização de Algoritmos

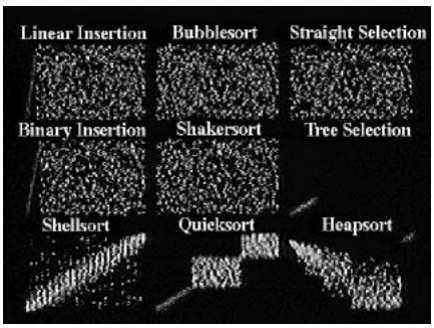
Segundo Kerren e Stasko (KERREN; STASKO, 2002) a visualização de algoritmos (ou animação) é a visualização do funcionamento em alto nível do comportamento de um algoritmo através da abstração de seus dados e operações, sendo que segundo Price, Baecker e Small (1993) a VA se enquadra como uma categoria mais específica de visualização de software, que engloba outros usos da visualização em softwares, como para o auxílio de depuração de programas, análise do fluxo de dados lidados pelo sistema, e outros usos.

2.2 Evolução da Visualização de Algoritmos

A importância de representações visuais para a compreensão do funcionamento de programas foi reconhecida à muito tempo, sendo que em já em 1947, Goldstein e von Neumann demonstraram a utilidade de fluxogramas (PRICE; BAECKER; SMALL, 1993) para o auxilio da compreensão do funcionamento de programas.

Apesar dessa constatação já em 1947, o início da utilização da visualização computacional para o ensino de ciência da computação geralmente tem seu inicio atribuído no começo da década de 80, com o vídeo "Sorting out Sorting"' criado por Baecker e Sherman em 1981, que consistia em um vídeo de 30 minutos contendo a visualização de 9 algoritmos de ordenação (Figura 2), sendo que sua criação a posterior distribuição teve um grande sucesso (BAECKER , 1998), e inspirou o desenvolvimento da área, sendo que diversos sistemas de visualização foram criados posteriormente.

Figura 2 – Cena do *Sorting out Sorting* onde se compara a velocidade de ordenação dos algoritmos.

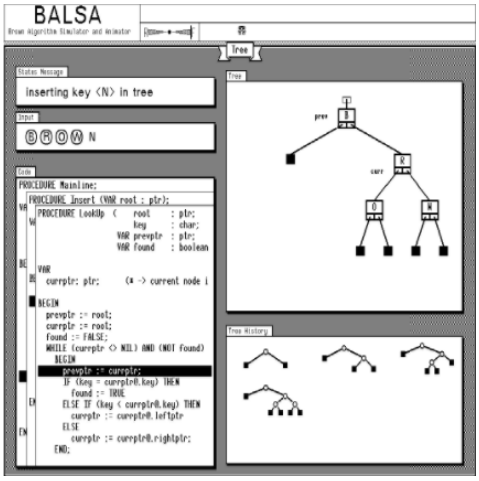


Fonte: (BAECKER, 1998)

Um dos primeiros sistemas de VA criados foi o BALSA (BROWN; SEDGEWICK, 1984), desenvolvido na universidade Brown, é um sistema integrado feito para realizar a "ani-

mação" de programas, sendo que ele permitia que dois tipos diferentes de usuários utilizassem o sistema, sendo que um deles poderia utilizar as ferramentas oferecidas para desenvolver animações próprias, que o outro tipo de usuário, sem precisar do conhecimento técnico de desenvolvimento no sistema, poderia visualizar e interagir com as animações através das três principais ferramentas disponibilizadas pelo sistema; uma tela de display que podia ter diversas visões de aspectos variados dos algoritmos, como as estruturas de dados, o pseudocódigo, e outras funções (Figura 3), um interpretador e um shell de comandos.

Figura 3 – BALSA.

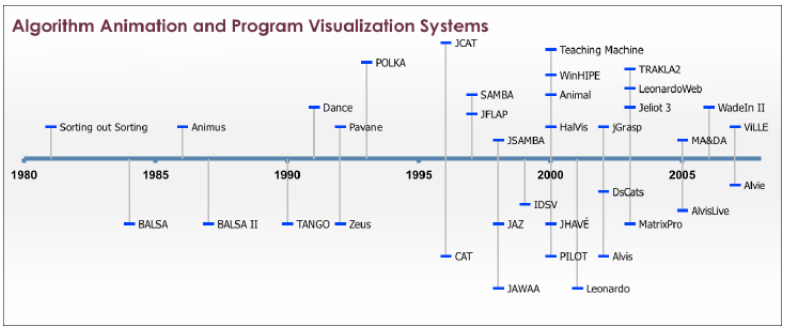


Fonte: (WARENDORF; HSU; SEAH, 1998)

O BALSA foi utilizado extensivamente na universidade Brown para o ensino de conceitos fundamentais da ciência da computação e também para depuração e pesquisas referentes à análise de algoritmos (SHAFFER et al., 2010), sendo que o sistema continuou a ser aperfeiçoado e teve uma segunda versão lançada, o BALSA II (BROWN, 1988), que também oferecia suporte a utilização de cores e sons.

Após isso, surgiram diversos sistemas, como o Tango (STASKO, 1990), TangoX (STASKO, 1992) e muitos outros, sendo que alguns podem ser observados na Figura 3.

Figura 4 – Surgimento de sistemas de visualização de algoritmos



Fonte: (KARAVIRTA, 2009)

Os primeiros sistemas a desenvolvidos definiram muitas práticas que posteriormente foram adotadas pelos seus sucessores, porém, eles possuíam algumas características que dificultavam a sua disseminação, como a dificuldade de construir as visualizações e falta de compatibilidade entre sistemas (FOUH; AKBAR; SHAFFER, 2012), sendo que apesar de que os sistemas desenvolvidos no final dos anos 80 e no começo dos anos 90 terem procurado utilizar o X-Window como uma maneira de aumentar o acesso aos sistemas, já que o X-Window oferecia uma base para interfaces gráficas independente do dispositivo, não foi possível obter muito sucesso já que na época computadores capazes de utilizar o X-Window não eram muito acessíveis (FOUH; AKBAR; SHAFFER, 2012).

Os problemas a respeito da acessibilidade começaram a mitigar após a metade da década de 90, com a disseminação do acesso a internet, o rápido crescimento da World Wide Web e crescimento do Java, que na época dominou a implementação desses sistemas, sendo que segundo um estudo realizado por Shaffer sobre o estado do campo de estudo (SHAFFER et al., 2010) virtualmente todos os sistemas e visualizações da época foram implementadas utilizando Java, sendo ele utilizado em conjunto com tecnologias web, através da utilização de applets, ou como aplicações Java que necessitavam ser baixadas e instaladas na máquina virtual Java local.

Apesar da ampla utilização do Java no desenvolvimento de sistemas de VA por diversos anos, recentemente o Java vem perdendo espaço na internet para o HTML5 em conjunto com o JavaScript, sendo que devido a falhas de segurança e problemas relacionados a malwares, seu suporte tem sido removido dos browsers mais populares, devido a isso, a utilização dessa ferramenta para o desenvolvimento de sistemas que visam ter uma grande acessibilidade tornou-se contra produtivo (KARAVIRTA; SHAFFER, 2013).

Tendo isso em mente, diversos desenvolvedores passaram a criar ferramentas para o desenvolvimento de visualizações que utilizassem HTML5 e JavaScript, já que essas tecnologias tem o suporte de todos os principais browsers com grande consistência, aumentando assim a acessibilidade dos sistemas de VA desenvolvidos e também a compatibilidade dos mesmos através de diversas plataformas (KARAVIRTA; SHAFFER, 2013)

2.3 Eficácia pedagógica

Apesar de que os professores da área de Ciência da Computação terem uma visão predominantemente positiva a respeito do impacto da utilização de visualizações no ensino (NAPS et al., 2002), a eficácia da utilização de VA no ensino não é facilmente comprovada, sendo que diversos estudos falharam em achar algum impacto conclusivamente positivo na utilização de alguns sistemas (HUNDHAUSEN; DOUGLAS; STASKO, 2002), porém esses mesmos estudos revelaram que essa falta de impacto é devida a maneiras de implementação e das teorias de aprendizado utilizadas para o desenvolvimento desses sistemas, e não devido a uma falha

intrínseca das visualizações como um meio de ensino.

2.3.1 Sistemas apropriados para o ensino

Em 2002, Hundhausen, Douglas e Stasko realizaram uma meta-análise a respeito da eficácia de VA no aprendizado, sendo que foram analisados os resultados de 24 estudos experimentais a respeito do impacto da utilização de VA no ensino, sendo que como ainda não existe um consenso acadêmico a respeito de qual teoria de aprendizado é a mais apropriada para modelar o aprendizado humano, cada estudo foi analisado de acordo com a teoria de aprendizado adotada por seus autores. No campo da VA são utilizadas predominantemente 4 diferentes teorias de aprendizado para embasar a eficácia pedagógica de sistemas de visualização (HUNDHAUSEN; DOUGLAS; STASKO, 2002), a fidelidade epistemológica, dual-coding, diferenças individuais e o construtivismo cognitivo, sendo que segundo os resultados da meta-análise, ficou constatado que o construtivismo cognitivo é a teoria de aprendizado mais robusta, com a maior previsão de resultados significativos (77%) e também a maioria dos resultados não-significativos (60%), a teoria construtivista será descrita brevemente a seguir.

2.3.2 Construtivismo Cognitivo

A teoria do construtivismo cognitivo se baseia na suposição de que o conhecimento não é uma representação absoluta de algum aspecto objetivo da realidade, mas sim uma estrutura mutável que é construída por cada indivíduo conforme se é adquirido novas experiências ao interagir ativamente com o ambiente, e sendo assim, essa teoria afirma que a forma mais eficaz de adquirir novos conhecimentos é através da realização de atividades que favoreçam um envolvimento mais ativos dos estudantes, em contraste com atividades mais passivas, como a leitura (RESNICK, 1989).

Os sistemas de VA que adotaram essa teoria como base procuraram focar não simplesmente na qualidade das suas animações, já que o ato de simplesmente visualizar o funcionamento de um algoritmo não é uma tarefa que envolve ativamente os alunos, mas também na construção de ferramentas que dão suporte a visualização e oferecem atividades que forçam os alunos a interagirem de uma forma mais ativa com o sistema, sendo que os sistemas que criaram formas de estimular a participação ativa dos estudantes obtiveram um resultado melhor quando comparados a sistemas que não empregassem formas ativas de interação (HUNDHAUSEN; DOUGLAS; STASKO, 2002).

2.3.3 Taxonomia de Engajamento

Conforme foi se tornando evidente que o nível de interação com as ferramentas de aprendizado era uma característica essencial para o sucesso pedagógico delas, surgiu a necessidade de uma nomenclatura adequada para tratar dos diferentes tipos de interações possíveis,

sendo que (NAPS et al., 2002) definiu uma taxonomia composta de 6 possíveis níveis de engajamento: sem visualização, visualizando, mudando,respondendo, construindo e apresentando.

2.3.3.1 Sem visualização

Onde não se é utilizado nenhum tipo de ferramenta de visualização.

2.3.3.2 Visualizando

A categoria de visualização pode ser considerada a parte fundamental dos diferentes níveis de interação possíveis, já que todas as outras categorias de interação englobam de alguma maneira a visualização, sendo que o simples ato de visualização pode ser considerado o nível mais baixo de interação possível com sistemas que empregam VA, a maioria das ferramentas de ensino para VA se encontram nessa categoria (NAPS et al., 2002)

2.3.3.3 Mudando

Com esse nível de interação, é possível e encorajado que os usuários realizem mudanças no algoritmo, podendo analisar o funcionamento da VA em diferentes situações, uma das formas mais comuns de possibilitar que o sistema ofereça esse nível de interação é permitir que o usuário insira diferentes inputs.

2.3.3.4 Respondendo

Nessa categoria é apresentado ao usuário perguntas referentes as visualizações apresentadas, portanto as visualizações se tornam um recurso para ser utilizado com o intuito de responder as perguntas, promovendo assim um nível de engajamento maior com o sistema.

2.3.3.5 Construindo

Nessa categoria de interação, os usuários do sistema devem construir as suas próprias animações sobre o algoritmo estudado, sendo que isso não implica necessariamente em requisitar que os usuários implementem o algoritmo na forma de código, já que por se tratar de um sistema que utiliza visualização, o algoritmo pode ser construído utilizando primitivas gráficas.

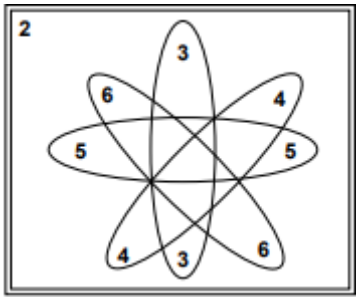
2.3.3.6 Apresentando

A última categoria definida por Naps (2002) et al consiste em apresentar a VA para uma audiência para feedback e discussão, sendo que as visualizações não precisam necessariamente terem sido criadas pelo apresentador.

Um ponto importante é que esses diferentes níveis de interação não formam uma estrutura hierárquica simples, onde um nível superior requisita necessariamente dos outros (excluído

o nível da visualização, que é necessário para realizar as formas de interações subsequentes), sendo assim possível desenvolver sistemas que englobam diferentes combinações de níveis de interações, como mostrado pelo Diagrama de Venn da figura 5.

Figura 5 – possíveis combinações entre os diferentes níveis de interação



Fonte: (NAPS et al., 2002)

2.4 Outras técnicas para construção eficaz

Além da constatação que o nível de interação com as VA influencia de sua eficácia pedagógica, existem outros fatores que contribuem para que um sistema seja eficiente para o ensino, como a utilização das cores e a forma de realizar o controle das visualizações.

2.4.1 Cores

Segundo Khuri (2001), a utilização de cores nas visualizações deve ser realizada com cuidado, pois apesar de que cores terem sido para aprimorar informações disponíveis em preto e branco, já que a utilização deles permite passar mais informações sem a necessidade de utilizar outras formas. Não foi constatado nenhuma diferença na habilidade do usuário em interpretar a informação, sendo que a utilização sem um cuidado planejado de cores pode ocasionar um excesso de informação e confundir o usuário (KHURI, 2001), tendo isso em vista Khuri recomenda a utilização de no máximo cinco cores(mais ou menos 2).

2.4.2 Controles da Visualização

Os sistemas de VA utilizam primariamente dois tipos de controles para a visualização das animações, sendo eles o de velocidade, que define o quão rápido se itera pela animação, sendo que comumente a visualização fica presa em um loop de execução, e o controle manual dos passos da visualização, sendo que nesse caso o usuário define quando que a animação mostrará o próximo passo do algoritmo, sendo que dentre esses dois tipos de controle, o controle manual sobre os passos da VA possui um efeito positivo maior para o entendimento dos algoritmos (SARAIYAN, 2002).

3 Métodos e Materiais

3.1 Materiais

A seguir, são descritos as principais tecnologias utilizadas para o desenvolvimento do projeto.

3.1.1 JSAV

JSAV é uma biblioteca para o auxilio do desenvolvimento de visualizações voltadas para aplicações Web, sendo uma das únicas do gênero até o momento (KARAVIRTA; SHAFFER, 2013). A JSAV foi escolhida para a implementação do projeto devido a atender as principais características necessárias para o desenvolvimento de um sistema de VA eficaz, e sendo assim consiste na principal tecnologia empregada para o desenvolvimento desse trabalho, dando suporte ao desenvolvimento das visualizações oferecendo diversas primitivas gráficas geradas com SVG, bem como um suporte nativo ao display de pseudocódigo, e outras ferramentas, o processo de criação e controle das visualizações será abordado com maior profundidade na seção de desenvolvimento. A descrição da utilização da biblioteca será abordada em mais detalhe na seção de desenvolvimento.

3.1.2 Nodejs e express

O NodeJS foi desenvolvido por Ryan Dahl em 2009, sendo que seu desenvolvimento foi impelido pelo desejo de criar uma maneira melhor de lidar com a comunicação entre servidor e cliente em tempo real, já que o servidor mais utilizado na época, o Apache HTTP Server tinha dificuldade em lidar com um grande número de conexões simultâneas (BELITSOFT.COM, 2017). Devido a dependência da JSAV no NodeJS, ele se tornou uma escolha natural para ser a ferramenta escolhida para realizar a lógica do servidor do projeto, já que evita a necessidade de utilizar um outro sistema de servidores web, como o Apache, e além disso, no começo de 2010 foi desenvolvido o gerenciador de pacotes npm, um dos maiores repositórios de código aberto do mundo, e sendo assim oferecendo uma gama extensa e variada de bibliotecas para o desenvolvimento do projeto. O Express, criado em 2010 por TJ Holowaychuk é uma *framework*para o desenvolvimento de aplicativos Web utilizando o NodeJS, sendo uma das *frameworks* mais utilizadas em conjunto com o NodeJS devido a sua versatilidade, já que o Express não define uma estrutura padrão para as suas aplicações.

3.1.3 EJS

EJS, ou *Embedded* JavaScript (EMBEDDEDJS.COM, 2017), é uma linguagem de modelagem simples que permite a geração de marcações de HTML com JavaScript simples, facilitando assim a geração de páginas que serão disponibilizadas pelo servidor, sendo que seu funcionamento gira em torno da utilização de *tags* especiais que permitem a execução de JavaScript sem a utilização de *tags* de HTML, permitindo realizar o controle da estrutura do HTML de forma mais fácil. A figura 6 demonstra a utilização do EJS em conjunto com uma página HTML.

Figura 6 – Utilização do EJS para estruturação de HTML.

Fonte: Elaborada pelo autor.

3.1.4 MongoDB

O MongoDB é um banco de dados não relacional criado em 2009 pela empresa 10Gen como um projeto open-source, sendo atualmente o banco de dados não relacional mais utilizado (ENGINES.COM, 2017), possui uma crescente comunidade de desenvolvedores, e foi escolhido como o banco de dados para o projeto devido a ser comumente utilizado em conjunto com aplicações que empregam NodeJS, sendo que diversos pacotes já foram desenvolvidos e disponibilizados através do npm.

3.2 Metodologia

Para a realização dos objetivos propostos, foi realizada uma revisão bibliográfica com o intuito de definir as atuais melhores práticas para o desenvolvimento de VA de qualidade, bem como um estudo a respeito da história dos sistemas e tecnologias relacionadas já implementadas. Após isso, foi realizado um estudo a respeito das tecnologias escolhidas para o desenvolvimento do projeto, com o intuito de compreender de forma eficaz como utiliza-las para atingir os objetivos propostos pelo projeto. Em sequência, o enfoque foi a definição dos requisitos e do projeto geral da ferramenta, tendo em vista a necessidade de atender as práticas para o desenvolvimento de visualizações pedagogicamente eficazes. Por fim, o projeto entrou em fase de implementação, com o desenvolvimento do *front-end* em conjunto das visualizações e o *back-end* com as perguntas e a lógica do servidor.

4 Descrição do projeto

4.1 Projeto geral da ferramenta

Nesse trabalho, foi desenvolvida uma aplicação web para o auxilio do ensino de inteligência artificial com a utilização da VA, demonstrando visualmente o funcionamento de diversos algoritmos. A aplicação, além das visualizações, também possui um sistema de perguntas e respostas, contendo perguntas formuladas pelo autor, mas também dando suporte para que os usuários da aplicação submetam e editem suas próprias perguntas, com o intuito de oferecer liberdade para os professores que utilizem o sistema definirem quais questões são mais adequadas para cada visualização em específico.

A aplicação foi desenvolvida no ambiente NodeJS utilizando o *framework* Express, que utiliza a linguagem JavaScript e o banco de dados MongoDB para a parte referente ao servidor. Para o desenvolvimento das visualizações, foi utilizado a JSAV, uma biblioteca que da suporte para o desenvolvimento de VA em JavaScript, já para a parte referente ao cliente foi utilizado o framework SemanticUI, para auxiliar no design da aplicação, a biblioteca de JavaScript jQuery, e as linguagens HTML5 e JavaScript.

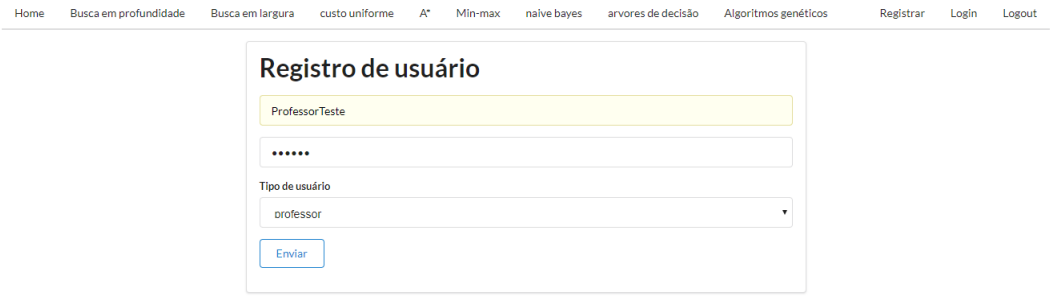
4.2 Cadastro e login

Como a aplicação visa dar suporte ao ensino de Inteligência Artificial não somente através da análise das visualizações, mas como também através de perguntas referentes a elas, além de possibilitar o acesso as visualizações e das perguntas pré-definidas para usuários não cadastrados, foi implementado um sistema de cadastro para poder dar aos usuários cadastrados a possibilidade de criar e responder perguntas diferentes das pré-definidas, sendo que o usuário pode se cadastrar como professor ou aluno.

4.2.1 Cadastro de um professor

Ao se cadastrar como professor, o usuário obtém acesso as opções de criar, editar e excluir perguntas referentes a quaisquer algoritmos, inclusive as que já são pré-definidas na aplicação, o usuário também obtém acesso aos perfis dos outros usuários que se cadastraram como seus alunos, podendo assim visualizar suas respostas e corrigi-las da forma que achar mais apropriada. A figura 7 exibe a tela de cadastro de um professor.

Figura 7 – Tela de cadastro de Usuários - Professor.

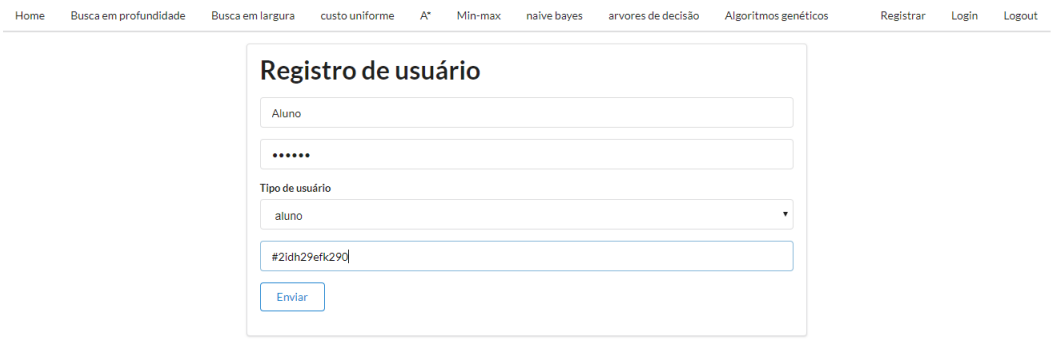


Fonte: Elaborada pelo autor.

4.2.2 Cadastro de um aluno

Ao se cadastrar como aluno, o usuário pode especificar a identificação de um professor, se tornando assim um aluno do mesmo e assim tendo acesso as perguntas definidas por ele, sendo assim as suas respostas serão corrigidas pelo professor e não pelo sistema de correção automático implementado. A figura 8 mostra a tela de cadastro, sendo que a opção de estudante foi selecionada, possibilitando assim ao aluno especificar seu professor através da chave do mesmo.

Figura 8 – Tela de cadastro de Usuários - Aluno.



Fonte: Elaborada pelo autor.

4.3 Sistema de perguntas e respostas

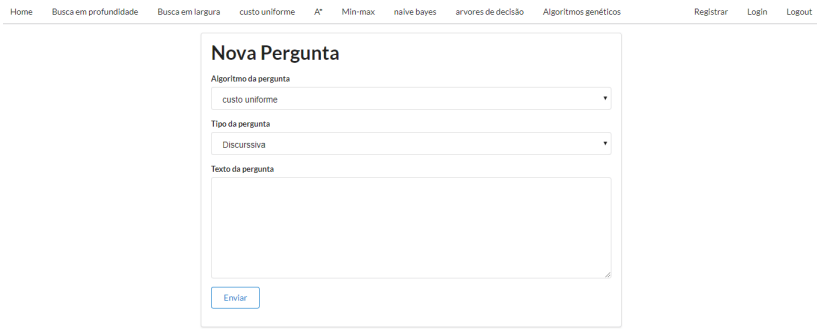
A sessão de perguntas e respostas implementada não foi a originalmente planejada, a biblioteca JSAV, que foi utilizada para a implementação das VA, possui uma implementação própria para perguntas e respostas, dando suporte a perguntas no estilo verdadeiro ou falso ou de múltipla escolha, podendo ser definidas em código para aparecer em momentos específicos das visualizações, porém, a implementação original não possibilita a edição e modificação das perguntas, estando ligadas de forma intrínseca ao código das visualizações, limitando assim a sua utilidade.

Como o projeto visa dar a opção de modificar e adicionar perguntas de acordo com a vontade do professor, foi desenvolvido um sistema próprio da aplicação de perguntas e respostas, dando suporte a perguntas discursivas, de múltipla escolha e verdadeiras ou falsas.

4.3.1 Adição de perguntas

O usuário cadastrado como professor possui acesso a capacidade de adicionar novas perguntas referentes a algoritmos específicos, definindo o tipo da pergunta, o texto da pergunta, suas alternativas e sua resposta correta, no caso de ser uma pergunta no estilo verdadeiro ou falso ou múltipla escolha. A figura 9 mostra a tela de adição de novas perguntas.

Figura 9 – Adição de uma nova pergunta.

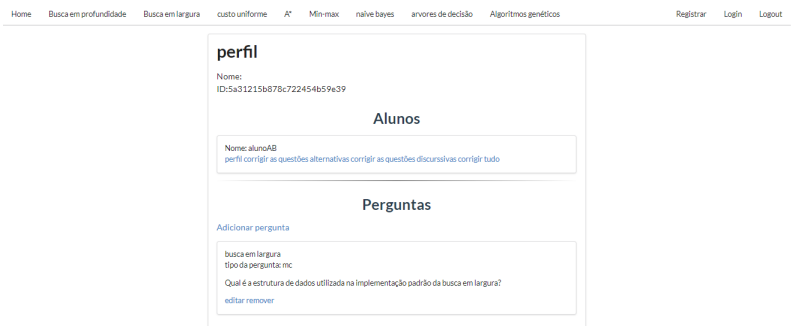


Fonte: Elaborada pelo autor.

4.3.2 Remoção de perguntas

Na tela de perfil do professor, pode-se encontrar todas as perguntas associadas a ele, sendo que para excluir uma pergunta em específico basta selecionar a opção remover (Figura 10).

Figura 10 – Remoção de uma pergunta.

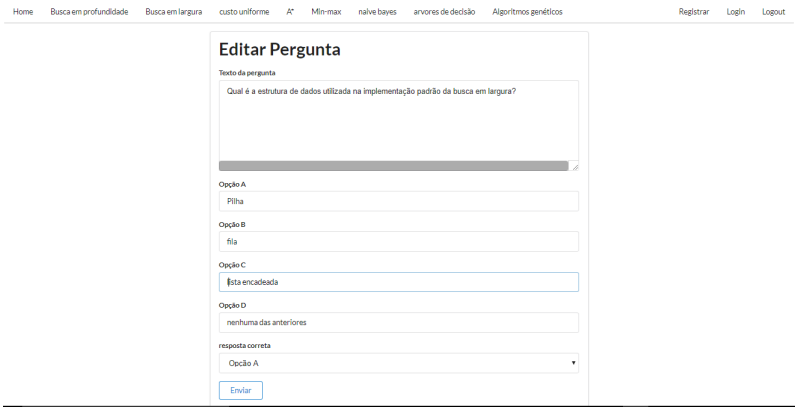


Fonte: Elaborada pelo autor.

4.3.3 Edição de perguntas

Também é possibilitado ao professor editar as perguntas adicionadas e as pré-definidas, podendo mudar o seu conteúdo, suas alternativas ou até mesmo excluí-las , A figura 11 ilustra a tela de edição de perguntas.

Figura 11 – Edição de uma pergunta.



Fonte: Elaborada pelo autor.

4.3.4 Correção das respostas

Dado que a aplicação pode ser acessada por usuários cadastrados e por usuários não cadastrados, a correção das perguntas foi implementada de duas formas diferentes, caso o aluno tenha se inscrito com um professor, ou caso não tenha um professor.

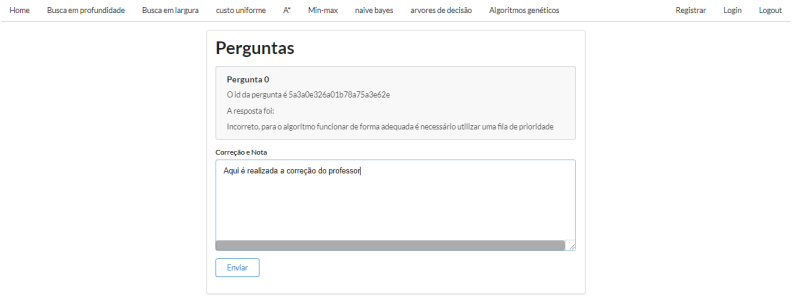
4.3.4.1 Usuário sem professor

Ao acessar a página das perguntas referentes ao algoritmo desejado, o usuário pode responder as perguntas disponíveis e receber automaticamente a correção, como ilustrado pela figura , sendo que como o usuário não possui um professor cadastrado, ele terá a opção de responder apenas perguntas do tipo verdadeiro ou falso ou de múltipla escolha.

4.3.4.2 Usuário com professor

Aos usuários com professor, o acesso a página referente ao algoritmo mostrará as perguntas definidas pelo professor, podendo ser do tipo verdadeiro ou falso, múltipla escolha ou discursivas, após responder as perguntas, fica a cargo do professor realizar a correção das mesmas, sendo que através da página de perfil o professor tem acesso a todos os seus alunos e pode realizar a correção das perguntas alternativas automaticamente. Já para as perguntas discursivas, ao decidir corrigi-las o professor será redirecionado a uma página para escrever a correção (Figura 12).

Figura 12 – Correção de uma pergunta discurssiva.



Fonte: Elaborada pelo autor.

4.4 Visualizações

A seguir, segue a descrição geral de como funciona a implementação das VA realizadas.

4.4.1 Funcionamento geral das Visualizações

As visualizações foram desenvolvidas utilizando a JSAV, que auxilia o desenvolvimento de VAs oferecendo além de primitivas de controle geral do funcionamento da animação, três tipos de objetos: estruturas de dados, primitivas gráficas e primitivas de código.

4.4.1.1 Controle das animações

O principal suporte dado pela biblioteca JSAV se da pela facilidade em que se é possível definir a sequência de eventos gráficos que ocorrem e como eles são gerados, sendo que a JSAV aborda a criação de VA através de *slideshows*, que utilizam a função *.step()* para delimitar a exibição das estruturas de dados e outros elementos gráficos a passos pré-definidos, possibilitando assim que o usuário da visualização itere sobre esses passos e assim visualize o funcionamento do algoritmo de acordo com o design do autor.

O usuário percorre a visualização através da interação com o elemento DOM *jsavcontrols* (Figura 13), que pela implementação nativa da biblioteca mostra 4 botões que permitem iterar sobre os passos definidos na programação de VA, sendo que os botões correspondem as instruções voltar ao começo da visualização, avançar um passo, retroceder um passo e ir para o final da visualização. O código referente aos botões de controle foi expandido nesse projeto para mostrar 2 botões adicionais, um que permite iterar automaticamente sobre os passos e um que para a iteração automática.

Figura 13 – Controles das Animações.

Fonte: Elaborada pelo autor.

Este elemento mostra os botões para o controle da visualização, sendo que para explicar como eles funcionam é necessário primeiro abordar a principal primitiva de controle da visualiza eles podem voltar ao início da visualização, retroceder um passo, ir para o próximo passo, e ir para o fim da animação.

Os botões gerados pela *jsavcontrols* funcionam para controlar qual instância de *.step()* deseja-se mostrar atualmente, facilitando o acesso do usuário aos diferentes estados da animação.

4.4.1.2 Objetos

Todos os objetos definidos pela JSAV entram na hierarquia DOM da página, podendo ter sua posição absoluta ou relativa mudada, além de poderem ser mostrados ou escondidos da visualização através da utilização das funções *.show() e .hide()*, bem como terem seu estilo mudado através da utilização de CSS, permitindo assim um alto nível de customização nas animações.

4.4.1.2.1 Estruturas de dados

As estruturas de dados suportadas pela biblioteca são: vetores, matrizes, listas encadeadas, árvores, árvores binárias e grafos.

4.4.1.2.2 Primitivas gráficas

As primitivas gráficas que a JSAV oferece são geradas através de SVG, sendo elas texto, linhas, círculos, retângulos, polígonos, poliline e caminho geral, que é utilizado para criar formas arbitrárias de linhas e curvas.

4.4.1.2.3 Elemento de código

A JSAV define um elemento para o display de pseudocódigo de fácil uso, consistindo em um conjunto de linhas que podem ser iteradas em conjunto com a função .step() para serem destacadas e assim sinalizarem qual operação está sendo executada nos passos específicos da visualização.

4.4.2 Visualizações Implementadas

A seguir, segue uma breve descrição dos algoritmos implementados e como a sua visualização foi realizada.

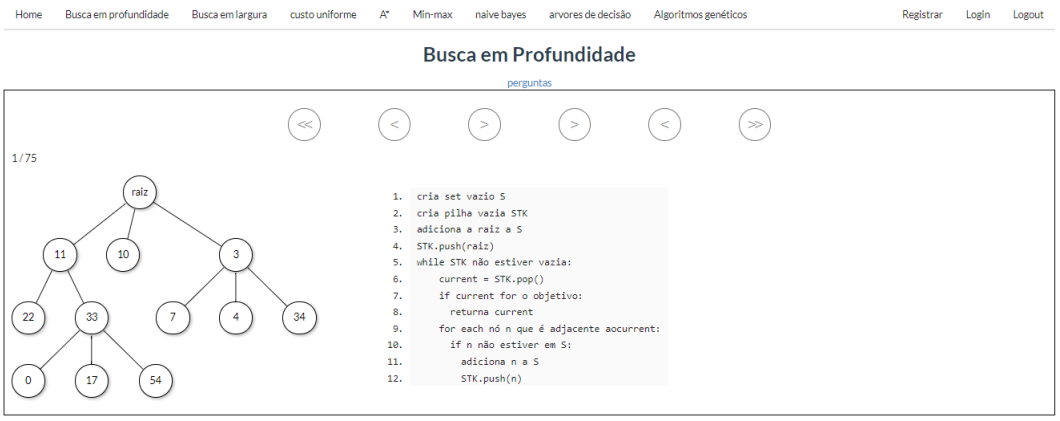
4.4.2.1 Buscas

Algoritmos de buscas são um mecanismo de resolução de problemas em geral, sendo aplicado no campo de Inteligência Artificial para resolver diversos problemas em que é necessário descobrir qual é a melhor maneira de atingir um estado desejado dado um estado inicial e um conjunto de possíveis ações (RUSSELL; NORVIG, 2003). Dada a sua importância na área, foram implementadas a animação de algumas buscas consideradas relevantes na área: a busca em profundidade, busca em largura, a busca de custo uniforme e a busca A*, sendo que a busca A* é diferente das demais por se tratar de uma busca informada, utilizando informações adicionais (heurísticas) a respeito do problema para chegar a solução.

4.4.2.1.1 Busca em Profundidade

O funcionamento geral do algoritmo consiste em percorrer a estrutura de dados (geralmente uma árvore ou um grafo) o máximo possível em um ramo antes de realizar o *backtracking* e seguir outro ramo, o estado inicial da VA implementada pode ser observado na figura 14 a seguir.

Figura 14 – Visualização da busca em profundidade.



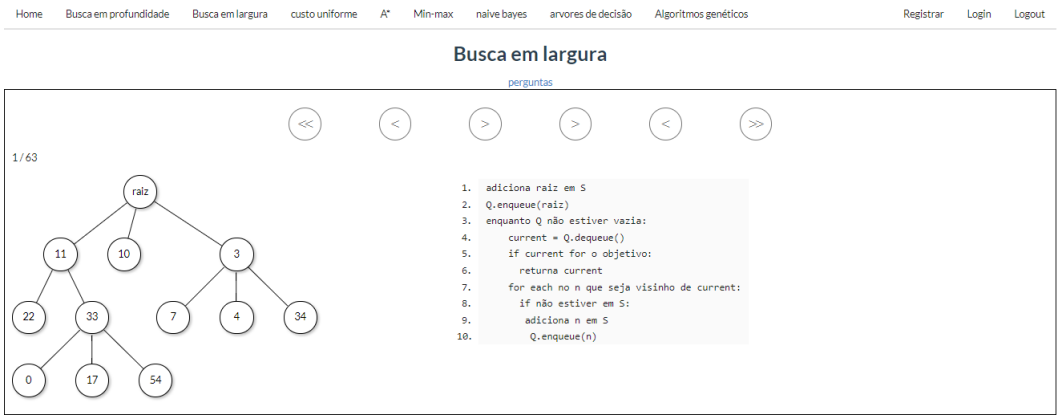
Fonte: Elaborada pelo autor.

4.4.2.1.2 Busca em Largura

A busca em largura funciona visitando a estrutura de dados pela ordem crescente de distância da posição inicial, percorrendo assim os possíveis valores em camadas.

A animação da busca em largura (Figura 15) possui as mesmas características utilizadas pela animação da busca em profundidade, já que do ponto de vista visual os dois algoritmos não possuem nenhuma distinção além da ordem dos nós visitados.

Figura 15 – Visualização da busca em largura.



Fonte: Elaborada pelo autor.

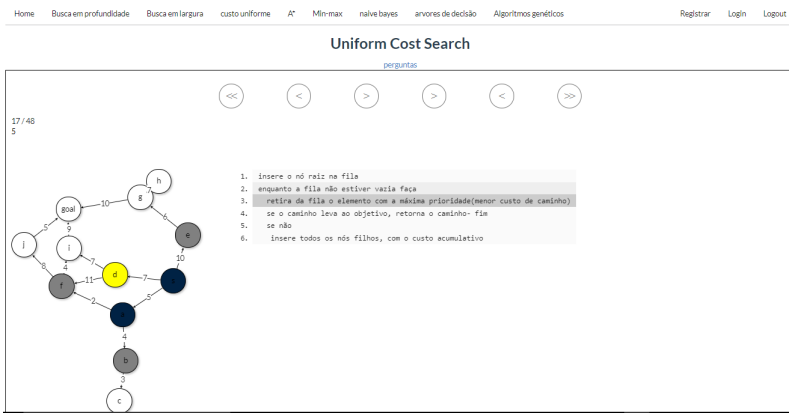
4.4.2.1.3 Busca de Custo Uniforme

A busca de custo uniforme é uma técnica de busca em que a ordem de visitação dos estados é determinada por aquele que possui o atual menor custo, sendo assim relacionada com a busca em largura, já que no caso do custo para chegar a um novo nó ser igual a profundidade, o algoritmo de custo uniforme funcionaria da mesma forma que a busca em largura.

Além de estar relacionado com a busca em largura, o algoritmo do custo uniforme é muito semelhante ao algoritmo de Dijkstra (FELNER, 2011), sendo que ao invés de achar o caminho de menor custo de todos os vértices até a raiz, o custo uniforme acha o caminho de menor custo da raiz até um vértice objetivo.

A implementação visual pode ser observada na figura a seguir, sendo que os nós cinzas são os nós em consideração de serem visitados, o nó azul é o nó atual e o amarelo é o no que já foi visitado, como pode ser observado na Figura 16.

Figura 16 – Visualização da busca de custo uniforme.



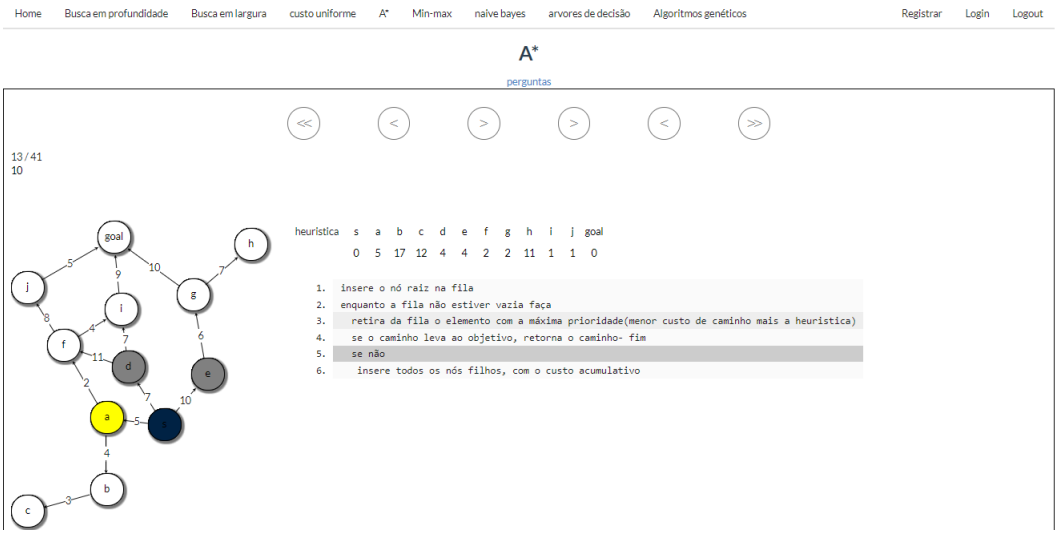
Fonte: Elaborada pelo autor.

4.4.2.1.4 A*

O algoritmo A* funciona de forma muito similar ao custo uniforme, porém para ser decidido qual nó será o próximo a ser visitado, o algoritmo faz uso de uma informação adicional além do custo até o nó destino, ele utiliza uma heurística que estima o qual será o custo dos nós até o nó objetivo, sendo assim ele define qual será o próximo nó calculando o menor valor de h(n) = Custo até N + custo estimado de N até Objetivo (RUSSELL; NORVIG, 2003), fazendo com que o algoritmo possa evitar de visitar nós com caminhos de baixo custo mas que não contribuam para atingir o objetivo.

A VA foi realizada de forma similar a busca de custo uniforme, porém com a adição de uma matriz mostrando os valores estimados dos para chegar até o nó objetivo (Figura 17).

Figura 17 – Visualização da busca A*.



Fonte: Elaborada pelo autor.

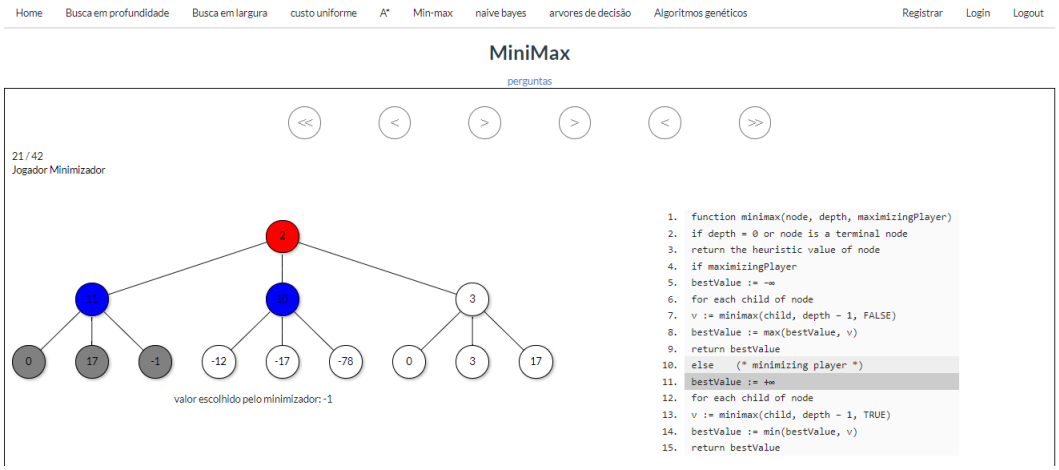
4.4.2.2 MiniMax

O algoritmo MiniMax é um algoritmo utilizado para realizar a tomada de decisões em cenários competitivos que envolvam dois competidores, sendo que o funcionamento do algoritmo em termos gerais normalmente gira em torno de uma função de utilidade capaz de determinar matematicamente o valor de um estado, sendo que um dos competidores deseja maximizar o valor atingido e o outro procura minimizar esse valor, sendo que o algoritmo toma como suposição que os competidores se alternam entre quem toma a decisão de qual será o próximo estado.

A VA foi realizada utilizando uma árvore com valores aleatórios, sendo que o jogador maximizador é o primeiro a realizar a decisão de qual será o próximo nó a ser visitado, sendo que ele é caracterizado pela cor vermelha, já o jogador minimizador é caracterizado pela cor azul. Ao percorrer os valores da arvore, a VA mostra em texto qual é o atual melhor valor

achado pelo algoritmo, até percorrer toda a árvore e retornar qual será o caminho escolhido pelo jogador maximizador (Figura 18).

Figura 18 – Visualização do algoritmo MiniMax.

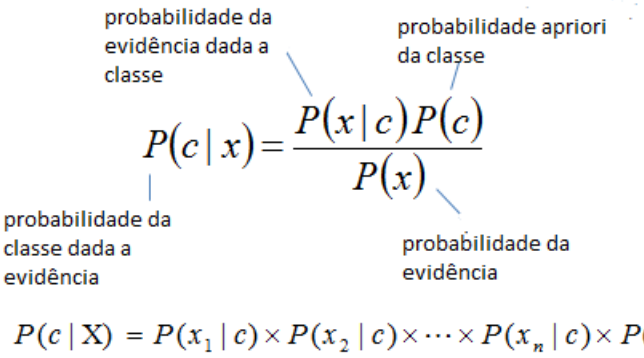


Fonte: Elaborada pelo autor.

4.4.2.3 *Naive Bayes Classifier*

No campo de aprendizado de máquinas, os *naive bayes classifiers* fazem parte da família de classificadores probabilísticos simples, que utilizam o Teorema de Bayes (Figura 19) em conjunto com a suposição de independência condicional entre atributos, ou seja, o algoritmo supõe que probabilidade de um dos atributos considerados não afeta a probabilidade de outro. Apesar de sua abordagem simplista e da suposição de independência, que frequentemente é incorreta, o algoritmo apresenta taxa de sucesso razoável em diversas áreas, como classificação de texto.

Figura 19 – O Teorema de Bayes e a fórmula simplificada utilizada no desenvolvimento.



Fonte: Elaborada pelo autor.

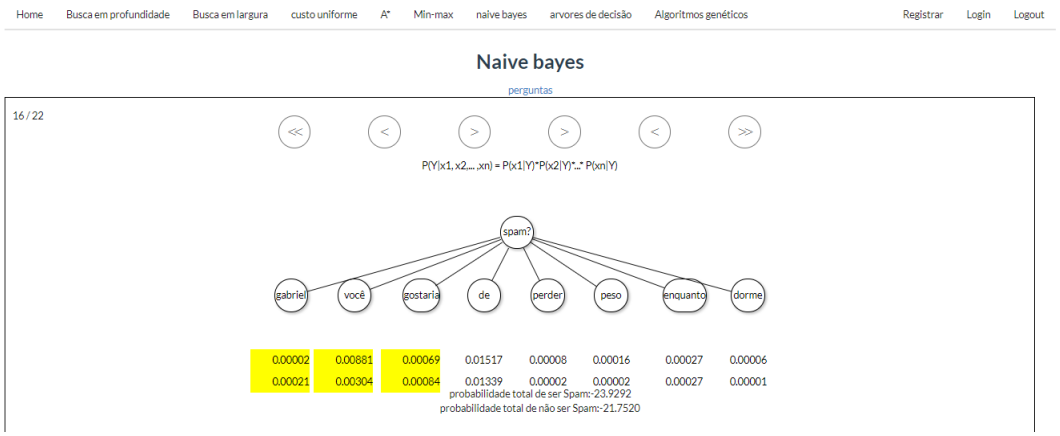
Com a suposição de independência das variáveis, o algoritmo funciona de forma bem direta, ele realiza o produtório das probabilidades individuais das evidências em cada um dos

casos possíveis de classificação, sendo que essas probabilidades são comumente obtidas através da análise de frequência. A classe que obtiver o maior valor final é dada como resposta pelo algoritmo, sendo que como o denominador não varia em nenhuma das classes, ele pode ser excluído do cálculo final para a geração da classificação (YANG; WEBB, 2002).

Apesar do seu funcionamento simples, do ponto de vista computacional, essa implementação direta do teorema de bayes pode ser aprimorada, pois já que se trata da multiplicação de probabilidades, se elas forem em grande número ou tiverem uma chance de ocorrer muito baixas. o cálculo da probabilidade de classificação corre o risco de ser arredondado para zero. Para excluir a chance de ocorrer esse erro, não foi realizado o produtório das probabilidades, mas sim o somatório do log delas, se aproveitando das propriedades dos logaritmos para realizar a soma dos valores, e não a multiplicação.

Na VA implementada, foi utilizado um exemplo de utilização dos naive bayes classifiers para a classificação de textos como spam ou não spam, sendo que a visualização começa explicando os conceitos necessários para aplicar o algoritmo, como o Teorema de Bayes, e além disso as probabilidades a priori das classes e das frequências das palavras utilizadas não são oriundas de casos reais, mas sim valores ilustrativos. Devido a simplicidade do algoritmo, não se viu necessidade de utilizar o display de pseudocódigo para permitir que o usuário acompanhe o funcionamento do algoritmo, tendo em vista que a partir do momento em que já se tem as tabelas de probabilidades necessárias para o aplicar o teorema de bayes, basta multiplicá-las para obter a resposta (Figura 20).

Figura 20 – Multiplicação das probabilidades das palavras



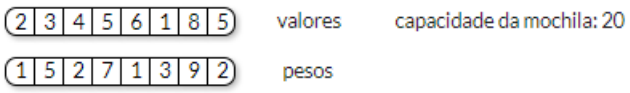
Fonte: Elaborada pelo autor.

4.4.2.4 Algoritmos Genéticos

Os algoritmos genéticos são uma classe de algoritmos de busca que visam imitar aspectos da biologia e da evolução com o intuito de encontrar soluções ótimas ou quase ótimas para problemas os quais não seria possível encontrar uma solução em um tempo viável.

A VA realizada sobre algoritmos genéticos aborda o problema da mochila, um clássico problema de otimização onde o objetivo é obter o maior lucro possível colocando o máximo de itens possíveis na mochila, mas sem ultrapassar o peso máximo dela (Figura 21).

Figura 21 – Especificações do Problema da Mochila utilizado no algoritmo genético



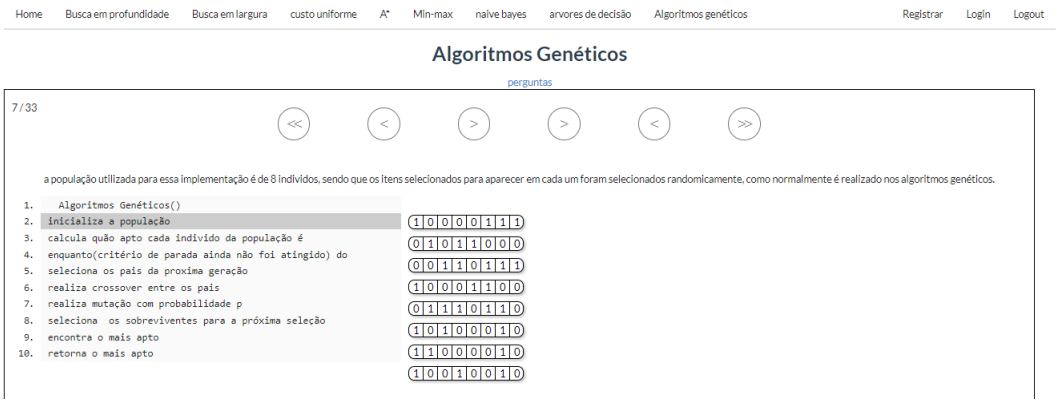
Fonte: Elaborada pelo autor.

O funcionamento geral de um algoritmo genético pode ser dividido em 6 fases: inicialização da população, calculo de aptidão, seleção dos parentes, realização do crossover entre os parentes, mutação e seleção de sobreviventes.

4.4.2.4.1 Inicialização da população

A população de um algoritmo genético representa as respostas candidatas do problema, sendo que a representação mais utilizada faz uso de vetores binários, onde o número um denota a presença da característica em questão e o zero a sua ausência, no problema abordado, cada posição do vetor representa um possível item de ser levado na mochila, contendo um custo(peso) e um determinado ganho. Na visualização implementada a população é de 8 indivíduos, sendo que seus valores iniciais são gerados automaticamente (Figura 22).

Figura 22 – Pseudocódigo e população inicial



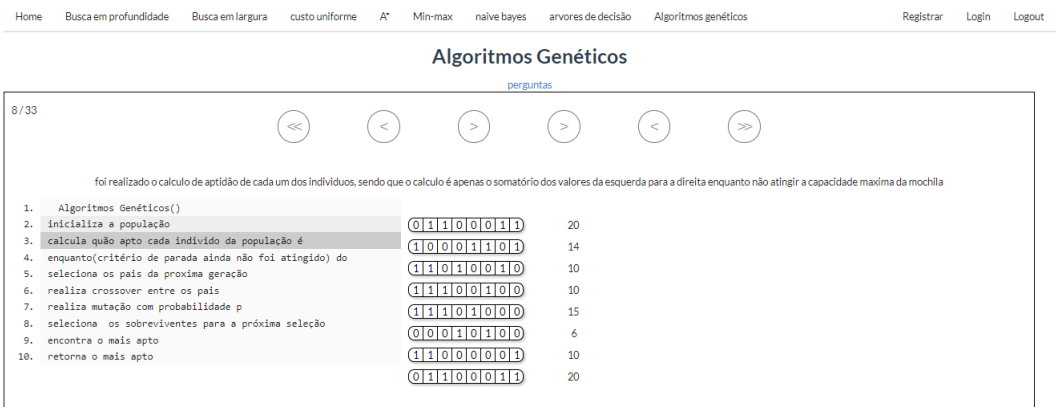
Fonte: Elaborada pelo autor.

4.4.2.4.2 Cálculo de aptidão

Nessa etapa é calculado o quão apto as soluções candidatas são, ou seja, quão bem elas resolvem o problema proposto, no caso do problema da mochila a função que define a aptidão é o somatório dos valores dos itens presentes no indivíduo da esquerda para a direita até que

o peso máximo da mochila seja atingido, a Figura 23 mostra os indivíduos da população com a sua respectiva aptidão calculada a direita do mesmo.

Figura 23 – Cálculo da aptidão de cada indivíduo da população

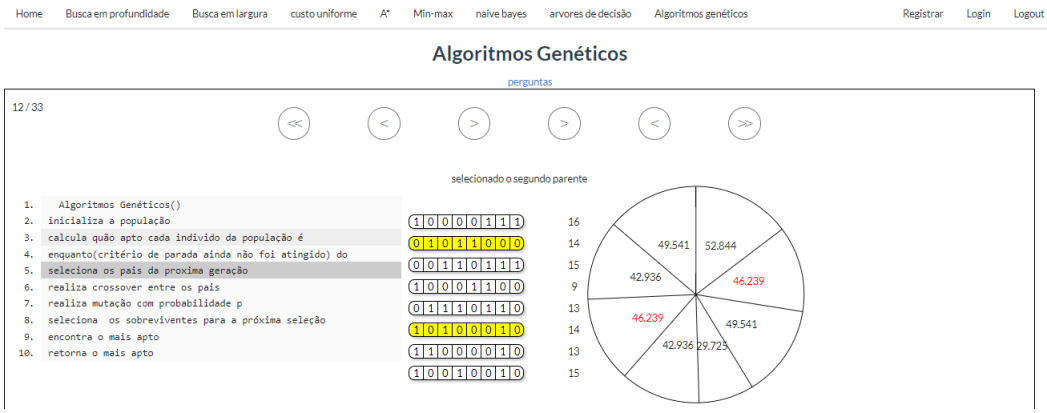


Fonte: Elaborada pelo autor.

4.4.2.4.3 Seleção dos parentes

A seleção de parentes implementada foi a seleção da roleta, onde a aptidão de cada indivíduo é representada em uma roleta, sendo que a quantidade de espaço ocupado por cada indivíduo corresponde a porcentagem que sua aptidão contribui para o total de aptidão da população, fazendo com que ao "girar" a roleta os indivíduos mais aptos possuem uma maior chance de serem selecionados como parentes para a próxima geração (Figura 24).

Figura 24 – Parentes da próxima geração



Fonte: Elaborada pelo autor.

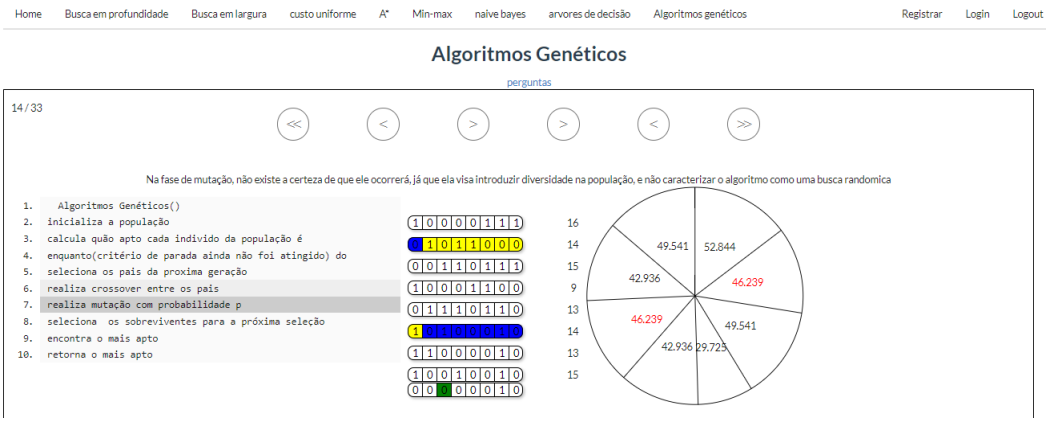
4.4.2.4.4 Crossover e Mutação

O crossover realiza uma mescla das características de ambos os parentes para gerar um novo indivíduo, sendo que o crossover implementado é o de ponto único, onde se é selecionado

aleatoriamente uma posição do vetor, um detalhe é de que como o ponto de crossover é selecionado aleatoriamente, é possível que a posição inicial ou final do vetor seja selecionada, fazendo com que o novo indivíduo receba apenas as características de um dos parentes.

A mutação é utilizada nos algoritmos genéticos para oferecer uma diversidade maior nas possíveis soluções, com o intuito de evitar que o algoritmo fique preso a pontos ótimos locais, sendo que seu valor normalmente não ultrapassa uma chance de 20% de ocorrer mutação, portanto essa foi a chance de mutação implementada no sistema, a mutação e destacada no indivíduo com a cor verde (Figura 25).

Figura 25 – *Crossover* e mutação realizada no novo indivíduo da população



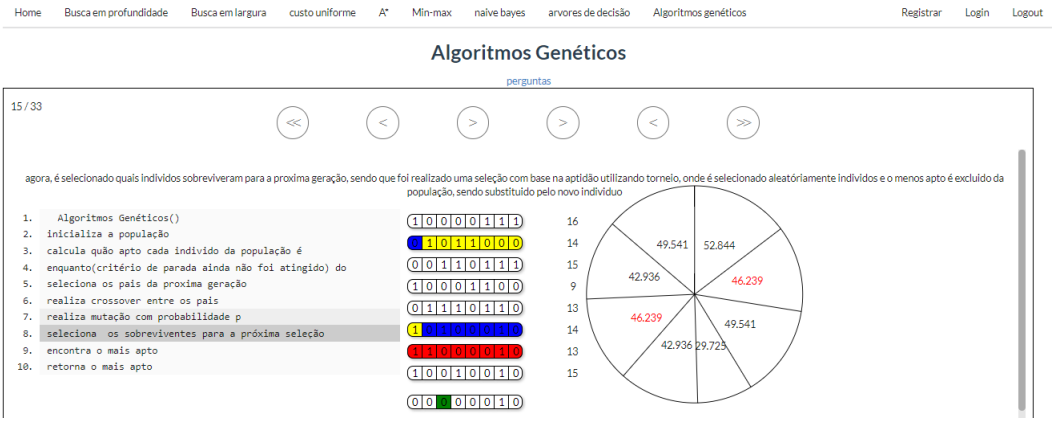
Fonte: Elaborada pelo autor.

4.4.2.4.5 Seleção de sobreviventes

A seleção dos sobreviventes faz alusão ao processo de seleção natural, sendo utilizado com o intuito de eliminar os indivíduos da população que provavelmente não contribuíram de forma eficaz para a otimização da resposta, a forma de seleção de sobreviventes adotada foi a de torneio, onde se é selecionado randomicamente uma parcela da população (4 indivíduos, na visualização) e o com o menor valor de aptidão é eliminado da população, sendo substituído pelo novo indivíduo criado, a Figura 26 mostra em vermelho o indivíduo que foi selecionado para ser eliminado da população.

Após a execução dessas fases, o algoritmo repete os processos de cálculo de aptidão até a seleção de sobreviventes enquanto o critério de parada definido não for atingido, como o objetivo da VA é apenas exemplificar o funcionamento dos algoritmos genéticos, o critério de parada adotado foi um número máximo de iterações igual a 3.

Figura 26 – Seleção do individuo que será eliminado da população



Fonte: Elaborada pelo autor.

4.4.2.5 Árvores de Decisão

As árvores de decisões são ferramentas utilizadas para modelar decisões e suas possíveis consequências, sendo utilizadas em áreas como pesquisa operacional e análise de decisões, porém, em aprendizado de máquina as árvores de decisões são uma ferramenta que é utilizada como um algoritmo de classificação. Sendo que as árvores podem ser construídas utilizando diversos algoritmos que agem sobre uma base de dados para determinar quais são as regras mais eficazes para particionar as entradas de uma maneira que diminuam ao máximo o nível de index Gini (uma métrica que mede o nível de desordem) dos nós sendo que a cada nó o algoritmo itera sobre os valores possíveis de entrada e constata se é possível particionar os dados de entrada no nó de uma maneira mais eficaz, sendo que se for possível diminuir a desigualdade o nó em questão gera 2 nós filhos, um nó para os elementos que deram a resposta positiva e o outro para os quais a resposta da pergunta é negativa para a pergunta.

A visualização do funcionamento das árvores de decisão foi realizada primeiramente explicando os conceitos utilizados, e após isso é construida a árvore, sendo que a árvore construída pode ser observada na Figura 27 a seguir.

Após a árvore ter sido concluída, as novas entradas apenas percorrem ela a vão seguindo o caminho de acordo com as respostas verdadeiras ou falsas a respeito das características da entrada, sendo que a nova entrada será classificada de acordo com a distribuição de das classificações no nó folha atingido (Figura 28).

Figura 27 – Árvore de Decisão

Fonte: Elaborada pelo autor.

Figura 28 – Caminho percorrido por uma nova entrada na árvore construída

Fonte: Elaborada pelo autor.

5 Conclusão

Conforme definido na introdução, o objetivo do trabalho consistia em desenvolver uma ferramenta para o auxilio do aprendizado de inteligência artificial utilizando a visualização de algoritmos. No trabalho foi pesquisado diversas tecnologias e sistemas que foram desenvolvidos ao longo de diversas décadas, sendo que foram estudadas diversas tecnologias para conseguir realizar o desenvolvimento do trabalho, além do estudo de diversos algoritmos de inteligência artificial para que fosse possível realizar as suas animações.

A ferramenta desenvolvida é completamente funcional, oferecendo além das visualizações um sistema de perguntas e respostas que permite um nível maior de engajamento com a ferramenta, sendo que o sistema posteriormente poderia ser aprimorado para contemplar outros níveis de interação propostos por Naps, sendo que a adição de mais atividades contribuiría para o aprimoramento do projeto.

Portanto, conclui-se que o projeto atingiu os seus objetivos principais, tendo sido criado uma aplicação web capaz de auxiliar o aprendizado de inteligência artificial com a utilização a VA.

Referências

BAECKER, R. Sorting out sorting – a case study of software visualization for teaching computer science. In: *Software Visualization – Programming as a Multimedia Experience*. [S.l.]: MIT Press Cambridge MA, 1998. p. 369–381. Citado na página 12.

BELITSOFT.COM. *PhP7 vs NodeJs*. 2017. Disponível em: <https://belitsoft.com/php-development-services/php7-vs-nodejs>. Citado na página 18.

BROWN, M. Exploring algorithms using balsa-ii. *IEEE Computer.*, v. 21, n. 5, 1988. Citado na página 13.

BROWN, M. H.; SEDGEWICK, R. A system for algorithm animation. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 18, n. 3, p. 177–186, jan. 1984. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/964965.808596>. Citado na página 12.

BYRNE, M. D.; CATRAMBONE, R.; STASKO, J. T. Evaluating animations as student aids in learning computer algorithms. *Computers Education*, v. 33, n. 4, p. 253 – 278, 1999. ISSN 0360-1315. Citado na página 10.

EMBEDDEDJS.COM. *EJS*. 2017. Disponível em: <https://embeddedjs.com>. Citado na página 19.

ENGINES.COM db. *database ranking*. 2017. Disponível em: <db-engines.com/en/ranking>. Citado na página 19.

FELNER, A. Position paper: Dijkstra's algorithm versus uniform cost search or a case against dijkstra's algorithm. In: BORRAJO, D.; LIKHACHEV, M.; LóPEZ, C. L. (Ed.). *SOCS*. AAAI Press, 2011. Disponível em: <http://dblp.uni-trier.de/db/conf/socs/socs2011-.htmlFelner11>. Citado na página 28.

FOUH, E.; AKBAR, M.; SHAFFER, C. A. The role of visualization in computer science education. *Computers in the Schools*, Routledge, v. 29, n. 1-2, p. 95–117, 2012. Disponível em: <https://doi.org/10.1080/07380569.2012.651422>. Citado na página 14.

HUNDHAUSEN, C.; DOUGLAS, S.; STASKO, J. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing.*, v. 13, p. 259–290, jun. 2002. Citado 3 vezes nas páginas 10, 14 e 15.

JENKINS, T. The motivation of students of programming. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 33, n. 3, p. 53–56, jun. 2001. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/507758.377472>. Citado na página 10.

KARAVIRTA, V. *Facilitating Algorithm Visualization Creation and Adoption in Education*. Tese (Doctoral Dissertation (Research Rep. No. TKK-CSE-A3/09)) — Helsinki University of Technology, 2009. Disponível em: <http://lib.tkk.fi/Diss/2009/isbn9789522481702-/isbn9789522481702.pdf>. Citado na página 13.

KARAVIRTA, V.; SHAFFER, C. A. Jsav: The javascript algorithm visualization library. In: *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer*

Science Education. New York, NY, USA: ACM, 2013. (ITiCSE '13), p. 159–164. ISBN 978-1-4503-2078-8. Citado 2 vezes nas páginas 14 e 18.

KEHOE, C.; STASKO, J.; TAYLOR, A. Rethinking the evaluation of algorithm animations as learning aids. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., Duluth, MN, USA, v. 54, n. 2, p. 265–284, fev. 2001. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1006-/ijhc.2000.0409>. Citado na página 10.

KERREN, A.; STASKO, J. T. *Chapter 1 Algorithm Animation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. 1–15 p. ISBN 978-3-540-45875-3. Citado na página 12.

KHURI, S. *Designing Effective Algorithm Visualizations*. 2001. Citado na página 17.

NAPS, T. L.; Rö, G.; ALMSTRUM, V.; DANN, W.; FLEISCHER, R.; HUNDHAUSEN, C.; KORHONEN, A.; MALMI, L.; MCNALLY, M.; RODGER, S.; VELáZQUEZ-ITURBIDE, J. A. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 35, n. 2, p. 131–152, jun. 2002. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/782941.782998>. Citado 3 vezes nas páginas 14, 16 e 17.

PRICE, B. A.; BAECKER, R. M.; SMALL, I. S. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing.*, v. 4, n. 3, p. 211–266, 1993. Citado na página 12.

RESNICK, L. B. Introduction. In: *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. [S.l.: s.n.], 1989. p. 1–24. Citado na página 15.

ROSENZWEIG, J.; BARTL, M. A review and analysis of literature on autonomous driving e-journal: Making-of innovation. October, 10 2015. Citado na página 10.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952. Citado 2 vezes nas páginas 27 e 29.

SARAIYAN, P. *Effective Features of Algorithm Visualizations*. Dissertação (Mestrado) — Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2002. Citado na página 17.

SHAFFER, C. A.; COOPER, M. L.; ALON, A. J. D.; AKBAR, M.; STEWART, M.; PONCE, S.; EDWARDS, S. H. Algorithm visualization: The state of the field. *Trans. Comput. Educ.*, ACM, New York, NY, USA, v. 10, n. 3, p. 9:1–9:22, ago. 2010. ISSN 1946-6226. Citado 3 vezes nas páginas 10, 13 e 14.

STASKO, J. Tango: A framework and system for algorithm animation. *Computer*, v. 23, n. 9, p. 27–39, 1990. Citado na página 13.

STASKO, J. Animating algorithms with xtango. *SIGACT News*, v. 23, n. 5, p. 67–71, 1992. Citado na página 13.

WARENDORF, K.; HSU, W. J.; SEAH, P. Y. Armvls-atomic reaction model visual language system-a new way of animating algorithms. In: *Proceedings of the World Conference on Educational Multimedia and Hypermedia*. [S.l.: s.n.], 1998. p. 1464–1470. ISBN 0-7803-3676-3. Citado na página 13.

YANG, Y.; WEBB, G. I. A comparative study of discretization methods for naive-bayes classifiers. In: *In Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*. [S.l.: s.n.], 2002. p. 159–173. Citado na página 31.