TuPy Online - Programação em Português com Visualização de Execução e Abstrações de Estruturas de Dados na Web

Giancarlo F. Roberto[1], **Fabiano S. Oliveira**[1]*,
Paulo Eustáquio D. Pinto[1]*, **Igor M. Coelho**[1]

[1]Instituto de Matemática e Estatística
Universidade do Estado do Rio de Janeiro (UERJ)
Rio de Janeiro – RJ – Brazil

giandroberto@gmail.com,

{fabiano.oliveira, pauloedp, igor.machado}@ime.uerj.br

Abstract. *There have been efforts to create tools that support the teaching of programming logic, such as program visualizers. One of them is Online Python Tutor which, despite having features for the representation of the state of variables for each step of the execution, does not allow convenient visualization of abstractions of more complex data structures, like graphs or trees. To address this problem, we present the TuPy Online tool, an adaptation of Online Python Tutor, proposing a lean syntax pseudolanguage (TuPy), commands in Portuguese and customizable display of data structures, being usable not only for program debugging but also for the preparation of courseware.*

Resumo. *Existem esforços para a criação de ferramentas de apoio ao ensino de lógica de programação, entre elas os visualizadores de programas. Um deles é o Online Python Tutor que, embora possua funcionalidades de representação do estado de variáveis a cada passo de execução, não permite a visualização conveniente de abstrações de estruturas de dados mais complexas, como grafos ou árvores. Para endereçar esse problema, apresentamos a ferramenta TuPy Online, uma adaptação do Online Python Tutor, propondo uma pseudolinguagem (TuPy) de sintaxe enxuta, comandos em português e exibição customizável para estruturas de dados, podendo ser utilizada tanto para depuração de programas quanto para preparação de material didático.*

1. Introdução

O estudo e o desenvolvimento de técnicas e ferramentas de suporte ao ensino da programação, sobretudo destinadas a estudantes da graduação, teve seu interesse aumentado durante a última década no âmbito da produção científica [Cunha et al. 2017]. Acredita-se que esse aprendizado é um esforço que deve ser capaz de se renovar em curtos intervalos de tempo para acompanhar o rápido ciclo evolutivo da tecnologia que ele tange. Esse processo de renovação pode ser ilustrado, por exemplo, pela adoção e o subsequente ganho de popularidade da linguagem Python para disciplinas de programação nas universidades do Brasil e do mundo [Barbosa et al. 2014, Guo 2014].

*Projeto parcialmente financiado por FAPERJ.

Além da escolha da linguagem de programação para o ensino, a facilidade de uso e o conjunto de funcionalidades do ambiente de desenvolvimento utilizado também são fatores que podem influenciar os resultados obtidos. Ambientes de desenvolvimento destinados a projetos de grande porte oferecem numerosas opções avançadas que, apesar de úteis para desenvolvedores profissionais, tornam mais difícil a navegação e o uso dessas ferramentas por parte do programador iniciante. Uma plataforma de aprendizado deve oferecer um escopo de funcionalidades apropriadamente limitado, tendo em vista que a facilidade de uso se dá através da simplicidade do campo de interação [Kölling 2010].

A ferramenta *Online Python Tutor* (OPT) [Guo 2013] foi projetada com o objetivo de providenciar um ambiente simples para visualização e depuração de pequenos programas escritos em Python, que fosse acessível através do navegador sem necessidade de instalação prévia. Desde então, recebeu um número considerável de usuários, tendo inclusive seu uso incentivado por cursos de graduação como o da University of California, Berkeley. Além disso, sua arquitetura foi projetada para ser extensível, viabilizando a gradual implementação do suporte a outras linguagens de programação, incluindo Javascript, C e Ruby. Entretanto, o OPT possui algumas limitações, como a ausência de recursos para se visualizar uma estrutura de dados de maneira personalizada, mais apropriada à abstração subjacente. Por exemplo, para depurar um programa em que um vetor é usado como um *heap*, é desejável que este possa ser exibido no formato de árvore binária, que evidencia as ligações entre nós pai/filho ao contrário de sua representação vetorial.

Este trabalho introduz o TuPy Online[1], uma ferramenta construída com base na arquitetura de OPT com os seguintes objetivos:

1. introduzir e utilizar uma pseudolinguagem (o TuPy) que seja o mais compacta e expressiva possível, de modo que com um número reduzido de palavras-chave seja possível expressar algoritmos gerais encontrados nos livros-texto;
2. servir ao público brasileiro, por ser uma linguagem definida com comandos em português, de forma a incluir a parcela da população que não é proficiente em outros idiomas, em particular o inglês;
3. permitir que algoritmos possam ser estudados com execução passo-a-passo e que estruturas de dados possam ser visualizadas de maneira apropriada conforme a abstração que representam.

2. Trabalhos Relacionados

Desde a concepção do sistema interativo BALSA [Brown e Sedgewick 1984], que gerava representações de alto nível de estruturas de dados de um programa Pascal, a área da visualização de *software* vem apresentando um número crescente de desenvolvimentos.

A plataforma de educação ViLLE começou como uma ferramenta para visualização passo-a-passo de programas elaborados por professores [Rajala et al. 2007]. Além de elaborar os programas-exemplo, o professor também podia configurar questões de múltipla escolha que eram exibidas ao aluno após passos específicos da visualização. O aluno, por sua vez, tinha a capacidade de modificar os programas recebidos para observar como as suas mudanças impactavam a execução. Os autores reforçam a necessidade do foco no raciocínio lógico em detrimento das particularidades das sintaxes das linguagens

[1]Disponível em https://gvirtu.github.io/tupy. Acessado em 30 mar. 2018.

de programação. A plataforma pode ser configurada para aceitar qualquer linguagem e oferece um pseudocódigo limitado que pode ser usado no visualizador.

O JavaTool permite que o aluno construa seu programa usando uma simplificação da sintaxe tradicional Java. Dessa forma, alavanca o ensino das disciplinas iniciais de programação através de animações da sequência de comandos executados e de explicações geradas automaticamente [Mota et al. 2008].

O uso de IDEs de baixa complexidade também é sustentado por [Noschang et al. 2014] que oferecem, através da ferramenta Portugol Studio, uma notação do Portugol[2] baseada em linguagens como C e PHP, além de bibliotecas que possibilitam a construção de programas com interfaces gráficas e reprodução de sons. Acompanha o *software* um extenso conjunto de demonstrações de funcionalidades, assim como exemplos de jogos de duas dimensões criados com ele.

Ao contrário das abordagens citadas até agora, que fazem uso de linguagens imperativas, a sintaxe encontrada em Potigol [Lucena e Lucena 2016] dispõe de construções multiparadigma, com ênfase no paradigma funcional.

Sorva et al. revisam mais de 40 sistemas de visualização de programa (incluindo OPT, ViLLE e JavaTool) e defendem o engajamento do usuário como fator fundamental para a eficácia de uma ferramenta didática de visualização [Sorva et al. 2013]. Propõem uma taxonomia bidimensional de engajamento que chamam de 2DET, com alicerce em taxonomias existentes [Naps et al. 2002, Myller et al. 2009]. Em uma dimensão, 2DET classifica o nível de envolvimento direto do aprendiz com a visualização, levando em consideração o controle que ele pode exercer sobre a ferramenta. A segunda dimensão diz respeito à autoria do conteúdo, classificado quanto à possibilidade de ser predefinido, parametrizável, modificável, ou elaborado pelo usuário. Ademais, os autores afirmam que existe um problema de disseminação no domínio do desenvolvimento de ferramentas de visualização. Segundo eles, em sua maioria os sistemas revisados aparentam ter sido protótipos de pesquisa descartados após sua elaboração ou após a condução de um estudo de avaliação, e comunidades de código aberto dedicadas ao trabalho em tais sistemas são "quase não existentes".

Vale destacar que os projetos Portugol Studio[3], Potigol[4] e OPT[5] seguem o modelo de código aberto na plataforma pública GitHub, estimulando o desenvolvimento colaborativo a longo prazo. Entretanto, na data de acesso aos respectivos repositórios, nenhum havia registrado um número expressivo de contribuidores, corroborando com as conclusões de Sorva et al.

As demais ferramentas mencionadas constituem visualizadores de programas. Existe uma outra classe de ferramenta, os visualizadores de algoritmos, que enfatiza a ilustração conceitual de certos algoritmos sem a necessidade de estarem rigorosamente alinhados a uma linguagem ou código, ao invés de evidenciar a interpretação da sequência de comandos de um programa para o acompanhamento de seu estado [Diehl 2007]. Um exemplo desse tipo de ferramenta é VisuAlgo [Halim 2015], que oferece uma coleção de

[2]Uma pseudolinguagem para a escrita de algoritmos em português.

[3]Disponível em https://github.com/UNIVALI-LITE/Portugol-Studio. Acessado em 13 fev. 2018.

[4]Disponível em https://github.com/potigol/Potigol. Acessado em 13 fev. 2018.

[5]Disponível em https://github.com/pgbovine/OnlinePythonTutor. Acessado em 13 fev. 2018.

algoritmos com visualizações animadas e individualizadas de acordo com as abstrações de suas estruturas, permitindo ainda que o usuário parametrize suas entradas.

Como será discutido adiante, o TuPy Online é uma iniciativa que busca unir a flexibilidade dos visualizadores de programa, mantendo a possibilidade de construir e depurar programas, à expressividade dos visualizadores de algoritmo, providenciando recursos para criar visualizações personalizadas.

3. Arquitetura do TuPy Online

Para o melhor entendimento de como o TuPy Online foi estruturado, dois componentes relevantes devem ser discutidos: o OPT e o interpretador da linguagem TuPy.

3.1. Online Python Tutor

Como foi mencionado, o OPT recebeu suporte a novas linguagens além de Python ao longo dos anos. Um aspecto de sua arquitetura que favoreceu tal implementação é o fato da visualização não ser construída diretamente a partir da interpretação do programa Python, mas sim da interpretação de uma estrutura de representação intermediária, a qual é gerada a partir da execução íntegra do programa. A questão de estender o suporte a novas linguagens se dá, portanto, pela implementação de um mecanismo para geração dessa representação intermediária. Para as linguagens atualmente suportadas, existem componentes separados encarregados de executar o código com restrições de privilégios e implementar alguma estratégia para produzir a estrutura desejada.

A representação utilizada é um arquivo JSON que descreve o rastro da execução, isto é, contém uma sequência de informações acerca do estado de variáveis e da pilha de ativação para cada passo da execução do programa. Como essa execução acontece integralmente de forma não interativa, o usuário pode, em posse do arquivo retornado, navegar livremente pela visualização da execução, em passos para a frente ou para trás.

No OPT, o rastro da execução de um programa Python é obtido através da análise de dados fornecidos pelo depurador `bdb` ao executá-lo. No TuPy Online, a concepção de um interpretador permitiu a integração de um módulo que acompanha a execução dos programas e é capaz de produzir o arquivo de rastro de execução no formato desejado.

3.2. O Interpretador da Linguagem TuPy

Foi utilizada a ferramenta ANTLR4 [Parr 2013] para que, a partir de uma gramática EBNF[6], houvesse geração de código para cumprir as etapas de análise léxica e sintática do interpretador, gerando uma árvore de sintaxe abstrata (AST).

Cabe ao interpretador, em seguida, percorrer a AST e registrar a evolução de estado do programa. Para isso, foram implementadas abstrações para a tabela de símbolos e a pilha de ativação, que contém as informações necessárias não somente à execução mas também à produção do rastro de execução.

4. Análise da Ferramenta

A interface do TuPy Online é totalmente baseada no navegador e herda diversos aspectos do OPT. Conforme mostra a Figura 1, a página principal consiste do campo de edição

[6]Uma extensão do Formato Backus–Naur (BNF) que, dentre outras características, possui notações que facilitam opcionalidade e repetições em regras.

de texto e dos botões de ação. A parte inferior da página também conta com um manual da linguagem TuPy e uma coleção de exemplos representativos de diferentes algoritmos e estruturas de dados. Durante a visualização, um painel de código situa-se à esquerda com destaque às linhas atual e seguinte da sequência de execução, junto aos botões de navegação. Assim como no OPT, é possível configurar pontos de parada por meio de um clique na linha de código desejada. Algumas funcionalidades do OPT, no entanto, ainda não estão disponíveis no TuPy Online, como colaboração em tempo real e a possibilidade de embutir um visualizador em outra página Web através de código HTML.

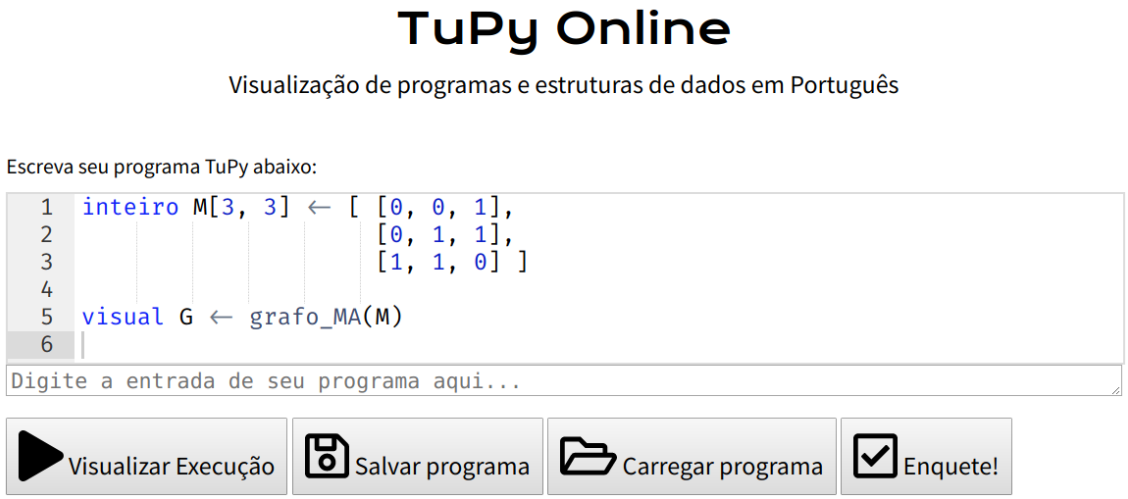


Figura 1. A interface de edição de código do TuPy Online.

Entre os diferenciais que foram implementados está a possibilidade de fornecer dados de entrada do usuário a serem lidos pelo programa, assim como botões para salvar e carregar o código-fonte de um arquivo de texto. O editor de código também foi adaptado para habilitar conveniências como preenchimento automático, expansão ou contração de blocos (*code folding*) e uma fonte[7] com suporte a ligaduras voltadas à programação, isto é, a capacidade de unir símbolos consecutivos de um código em um novo símbolo coerente automaticamente. A digitação dos símbolos correspondentes ao comparador de não-igualdade de operandos (`!=`), por exemplo, resulta na exibição do símbolo $\neq$. Com isso, espera-se que o código TuPy possa se aproximar visualmente de notações de pseudocódigo manuscrito. Além disso, foi disponibilizada uma versão que pode ser executada localmente, sem necessidade de conexão com a internet, minimizando problemas de escalabilidade oriundos do compartilhamento de recursos do servidor.

4.1. A Sintaxe da Linguagem TuPy

Assim como a documentação e todos os menus e textos do ambiente de desenvolvimento do TuPy Online, os comandos e as palavras reservadas presentes na linguagem estão em português, com suporte a caracteres *Unicode*, incluindo letras acentuadas. Essa deliberação se deu em vista do obstáculo linguístico encontrado em ferramentas existentes [Anido 2015]. A intenção é permitir o foco do programador aprendiz na estruturação

[7]Fira Code. Disponível em https://github.com/tonsky/FiraCode. Acessado em 18 fev. 2018.

e escrita do código, de forma a minimizar a carga cognitiva adicional oriunda da necessidade de mapear símbolos, vocábulos estrangeiros ou mnemônicos a um conjunto de comportamentos esperados.

Um dos objetivos originais do OPT era facilitar o processo de elaboração de conteúdo expositivo com o qual o docente precisa arcar, que requer esforço e é propenso a erros [Guo 2013]. Alinhada a esse objetivo, a linguagem TuPy foi derivada, após sucessivos refinamentos, de material que é aplicado a estudantes dos primeiros períodos do curso de Bacharelado em Ciência da Computação, e que foi elaborado por professores da Universidade do Estado do Rio de Janeiro. Possui, portanto, um conjunto restrito de funcionalidades que abrange o conteúdo tradicionalmente abordado por cursos introdutórios, incluindo comandos de repetição, suporte a recursividade e tipos compostos.

Em TuPy, o agrupamento de comandos em um mesmo bloco de código não necessita de marcadores de início e fim: foi adotado um estilo de sintaxe baseado em indentação similar ao encontrado em Python. O uso do ponto-e-vírgula como terminador de comando também não é obrigatório. Outras características importantes incluem atribuições paralelas, um sistema de tipos estáticos e a definição de tipos compostos com recursos do paradigma de orientação a objeto, como atributos e métodos, herança e polimorfismo.

Destinada ao uso integrado com o visualizador, a sintaxe de TuPy também permite ocultar certos elementos da sequência de visualização, incluindo valores de variáveis arbitrárias ou linhas de código inteiras. Dessa forma, um expositor pode destacar elementos específicos, filtrando funções, variáveis e estruturas auxiliares que não sejam essenciais ao entendimento de um algoritmo.

4.2. Visualização de Estruturas de Dados

Uma das limitações de OPT, nas palavras do autor, consiste na falta de visualização para "estruturas de dados sofisticadas". O projeto do TuPy Online buscou endereçar essa restrição providenciando uma biblioteca de funções para criar representações visuais extensíveis a partir de estruturas de dados.

É possível gerar visualizações de grafos (direcionados ou não), árvores, *heaps*, vetores, matrizes, listas encadeadas, pilhas e filas em qualquer passo da sequência de execução através de um único comando. A Figura 2 ilustra um caso de uso para essa funcionalidade: um grafo cujas arestas são descritas por uma matriz de adjacências. Note que a visualização OPT, embora visualize fielmente as variáveis que compõem o estado interno da estrutura (uma matriz, na forma de uma lista de listas), não sintetiza de forma compreensiva a abstração que ela representa (o grafo). O conjunto de funções de visualização que foram predisponibilizadas com a linguagem pode ser encontrado na Tabela 1. Em TuPy, uma mesma estrutura pode ser traduzida para diferentes formatos, como pode ser evidenciado também pela Figura 2, que ilustra a mesma matriz exibida como tabela e como grafo. As visualizações customizadas devem ser explicitamente especificadas pelo programador ao atribuir o resultado de uma das funções geradoras a uma variável, caso contrário a exibição OPT é mostrada por padrão.

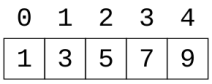
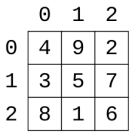
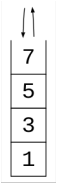
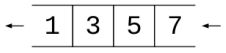
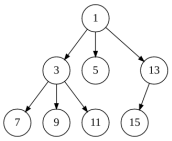
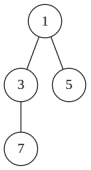
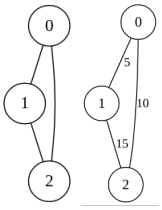
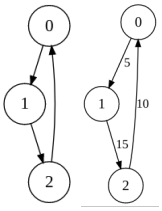
Estrutura	Função	Resultado
Vetor	`vetor(V)`	
Matriz	`matriz(M)`	
Lista Encadeada	`lista_encadeada(...)`	
Pilha	`pilha(V)`	
Fila	`fila(V)`	
Árvore	`árvore(...)`	
Heap	`heap(V)`	
Grafo	`grafo_MA(M)`	
	`grafo_LA(M)`	
	`grafo_valorado_MA(M)`	
	`grafo_valorado_LA(M)`	
Digrafo	`digrafo_MA(M)`	
	`digrafo_LA(M)`	
	`digrafo_valorado_MA(M)`	
	`digrafo_valorado_LA(M)`	

Tabela 1. Funções predefinidas para gerar visualizações de estruturas em TuPy. Foi utilizado V para simbolizar um parâmetro vetorial, M para um parâmetro em forma de matriz, e "..." para um parâmetro de tipo composto.

Internamente, as funções encapsulam lógica para percorrer as estruturas convertendo-as para descrições textuais, que são consumidas e transformadas em imagens durante a execução no navegador do usuário pelo Graphviz [Ellson et al. 2003], uma ferramenta para visualização de grafos e produção de diagramas estáticos descritos pela linguagem DOT [Gansner et al. 2015].

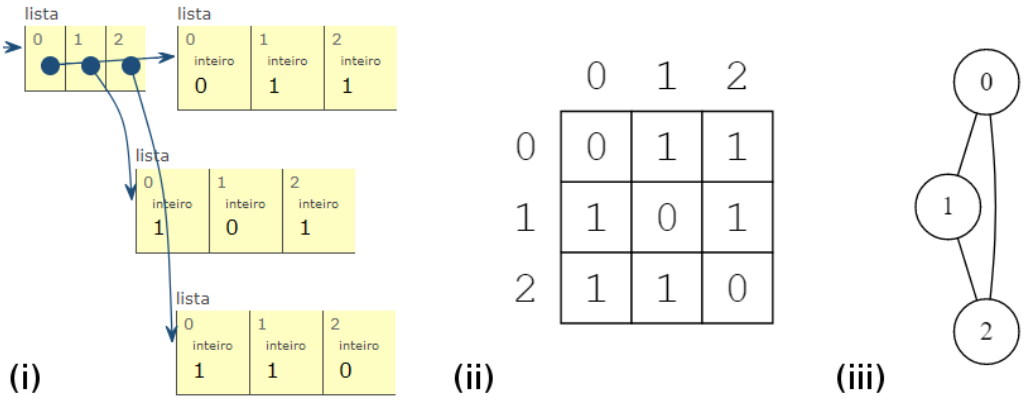


Figura 2. Visualização de uma matriz de adjacência de três formas no TuPy Online: a visualização tradicional do OPT (i), a visualização com a função de abstração `matriz` (ii), e com a função de abstração `grafo_MA` (iii).

O uso de descrições textuais na linguagem DOT, que possui uma extensa gama de opções para a customização dos formatos, das cores e do posicionamento dos elementos visuais, permite gerar visualizações parametrizadas pelo estado do programa. Como as funções de visualização geram programas DOT armazenados em cadeias de caracteres, é possível usar operações básicas como concatenação para manipular ou construir qualquer subprograma DOT válido em tempo de execução. Isso significa que, além das funções predefinidas para estruturas de dados convencionais, o programador pode criar novas funções capazes de exibir imagens com as características que desejar, complementando a visualização de forma integrada. Essa funcionalidade é observada na demonstração do problema das N rainhas [Weisstein 2002], incluído como um dos exemplos da ferramenta: uma função construída em TuPy é chamada pelo programa principal para criar iterativamente uma descrição de uma tabela cuja imagem resultante se assemelha a um tabuleiro de xadrez. Conforme se dá a execução do algoritmo, que usa a técnica de *backtracking* para posicionar N rainhas de xadrez em um mesmo tabuleiro N×N, as células do tabuleiro são enfatizadas através da mudança de cor na ordem de seus acessos pelo programa, e caracteres *Unicode* são usados para representar as peças atualmente posicionadas. Esse resultado pode ser visto na Figura 3.

5. Conclusões e Trabalhos Futuros

As finalidades do desenvolvimento do TuPy Online incluem a provisão de um ambiente para o ensino de programação que possa beneficiar tanto o professor quanto o aluno, oferecendo simplicidade e versatilidade suficientes para integrar o estudo de estruturas de dados a visualizações automáticas em um nível adequado de abstração. Tal recurso não substitui o depurador tradicional, mas pode ter um papel complementar na otimização de processos de construção de algoritmos até mesmo fora da sala de aula, tornando intuitivo o entendimento e a identificação de problemas em estruturas complexas. Adicionalmente, o

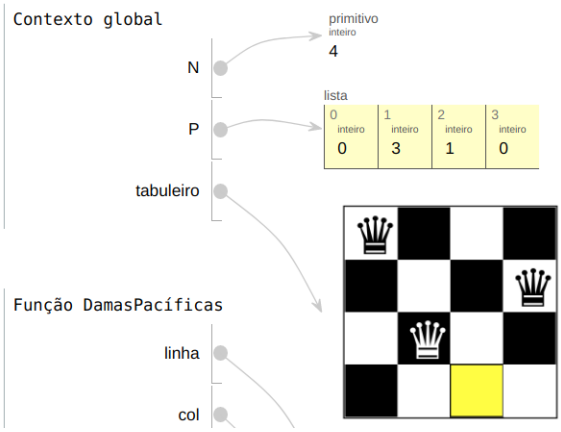


Figura 3. Tabuleiro de xadrez gerado por uma função TuPy durante a execução do algoritmo de backtracking para o problema *N* Rainhas (N=4)

projeto de código aberto, ao adaptar e integrar outras ferramentas consolidadas e testadas como OPT e Graphviz, visa promover um desenvolvimento estável e fomentar o reuso e o aprimoramento colaborativo de ferramentas existentes.

Entre os aspectos do projeto que podem ser futuramente aperfeiçoados estão: expressividade de mensagens de erro, ausência de explicações em linguagem natural para os comandos e interatividade na customização de visualizações.

Tem se fortalecido a crença de que os obstáculos para a adoção convencional de ferramentas de visualização não estão somente relacionados à abrangência de suas funcionalidades. Fatores como engajamento e facilidade de uso têm tido sua importância evidenciada na literatura. Espera-se que o TuPy Online possa, além de atender esses requisitos em algum nível, ser compatível com as necessidades curriculares e infraestruturais das universidades brasileiras, podendo assim ser uma contribuição catalisadora do ensino de algoritmos e estruturas de dados. Para que possa ser conduzido um estudo qualitativo da eficácia da ferramenta, o seu uso já está sendo implantado nos cursos introdutórios de computação do Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro.

Referências

Anido, R. (2015). Saci–ainda outro ambiente para o ensino de programação. *Anais do XXIII WEI - Workshop sobre Educação em Computação*.

Barbosa, A. d. A., Ferreira, D. Í., e Costa, E. B. (2014). Influência da linguagem no ensino introdutório de programação. Em *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 25, página 612.

Brown, M. H. e Sedgewick, R. (1984). *A system for algorithm animation*, volume 18. ACM.

Cunha, L. S., Tonetti, P., e Sanavria, C. Z. (2017). O ensino de informática no brasil: Uma análise da produção científica em eventos da sbc (2010–2014). *Anais do Computer on the Beach*, páginas 31–40.

Diehl, S. (2007). *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media.

Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., e Woodhull, G. (2003). Graphviz and dynagraph – static and dynamic graph drawing tools. Em *GRAPH DRAWING SOFTWARE*, páginas 127–148. Springer-Verlag.

Gansner, E., Koutsofios, E., e North, S. (2015). Drawing graphs with dot.

Guo, P. (2014). Python is now the most popular introductory teaching language at top us universities. *BLOG@ CACM, July*, página 47.

Guo, P. J. (2013). Online python tutor: Embeddable web-based program visualization for cs education. Em *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, páginas 579–584, New York, NY, USA. ACM.

Halim, S. (2015). Visualgo–visualising data structures and algorithms through animation. *OLYMPIADS IN INFORMATICS*, página 243.

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14:1–14:21.

Lucena, L. R. e Lucena, M. (2016). Potigol, a programming language for beginners. Em *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, páginas 368–368. ACM.

Mota, M. P., Pereira, L. W. K., e Favero, E. L. (2008). Javatool: Uma ferramenta para o ensino de programação. Em *Congresso da Sociedade Brasileira de Computação. Belém. XXVIII Congresso da Sociedade Brasileira de Computação*, páginas 127–136.

Myller, N., Bednarik, R., Sutinen, E., e Ben-Ari, M. (2009). Extending the engagement taxonomy: Software visualization and collaborative learning. *ACM Transactions on Computing Education (TOCE)*, 9(1):7.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., et al. (2002). Exploring the role of visualization and engagement in computer science education. Em *ACM Sigcse Bulletin*, volume 35, páginas 131–152. ACM.

Noschang, L. F., Pelz, F., e Raabe, A. e. a. (2014). Portugol studio: Uma ide para iniciantes em programação. *Anais do CSBC/WEI*, páginas 535–545.

Parr, T. (2013). *The definitive ANTLR 4 reference*. Pragmatic Bookshelf.

Rajala, T., Laakso, M.-J., Kaila, E., e Salakoski, T. (2007). Ville: a language-independent program visualization tool. Em *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88*, páginas 151–159. Australian Computer Society, Inc.

Sorva, J., Karavirta, V., e Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4):15:1–15:64.

Weisstein, E. W. (2002). Queens problem. *MathWorld–A Wolfram Web Resource*. Disponível em http://mathworld.wolfram.com/QueensProblem.html. Acessado em 22 fev. 2018.