O Ensino-Aprendizagem de Programação de Computadores: dificuldades e ferramentas de suporte

Bruno Siqueira da Silva[1], Toni Ferreira Montenegro[1]

[1]Instituto Federal Farroupilha (IFFar) *campus* São Borja
Otaviano Castilho Mendes, 355, 97670-000 – São Borja, RS – Brasil

{bruno.siqueira, toni.montenegro}@iffarropilha.edu.br

Abstract. *This article researched the factors that hinder the process of teaching and learning of beginner students computer programming in higher course in computing. For this, they present some tools and strategies identified by the literature to try to mitigate these problems. It was observed that the adoption of tools that allow graphic visualization algorithm behavior are more common in the researched cases, or not there is a direct use of source code, minimizing the syntactic and semantic aspect of the algorithm, but the road traveled by it to the desired solution. Thus, this study makes a contribution to the literature and the teaching of computing.*

Resumo. *Este artigo investigou os fatores que dificultam o processo de ensino e aprendizagem de programação de computadores de alunos ingressantes em curso superiores na área da computação. Para isso, são apresentados algumas ferramentas e estratégias apontadas pela literatura para tentar mitigar esses problemas. Observou-se que a adoção de ferramentas que permitem a visualização gráfica do comportamento do algoritmo são mais comuns nos casos pesquisados, ou seja, não ocorre uma exploração direta de código fonte, minimizando o aspecto sintático e semântico do algoritmo, mas sim o caminho percorrido por ele até a solução desejada. Dessa forma, este estudo deixa uma contribuição à literatura e aos docentes da área da computação.*

1. Introdução

O ensino e aprendizagem de programação constituem um enorme desafio para alunos e professores. Os elevados níveis de insucesso nas disciplinas introdutórias de programação, em qualquer grau e sistema de ensino, em qualquer parte do mundo, são tema de preocupação e alvo de variadas pesquisas (Charlton e Luckin, 2012; Ramos e Espadeiro, 2014). O ensino de programação tem como propósito conseguir que os alunos desenvolvam as suas capacidades, adquirindo os conhecimentos e competências necessárias para conceber programas e sistemas computacionais capazes de resolver problemas reais. Porém, a experiência tem demonstrado que existe, em termos gerais, uma grande dificuldade em compreender e aplicar certos conceitos abstratos de programação, por parte de uma percentagem significativa dos alunos que frequentam disciplinas introdutórias nesta área. Uma grande dificuldade reside na compreensão e, em particular, na aplicação de noções básicas, como as estruturas de controle, para a criação de algoritmos que resolvam problemas concretos. Essas dificuldades traduzem-se inevitavelmente em elevadas taxas de insucesso e desistência (Kulyk et al., 2007).

Para Ramos e Espadeiro (2014) é fundamental que alunos iniciantes desenvolvam habilidades para resolver problemas, boa capacidade de abstração e consigam aplicar o conhecimento adquirido fazendo uso de linguagens de programação, ou seja, transformar processos abstratos em concretos, através de produtos de *software*

que possibilitem melhor compreensão dos conceitos abordados e maior interação entre aluno-objeto de trabalho (neste caso, o computador). No entanto, é necessário que docentes mantenham o olhar atento ao processo de ensino e aprendizagem dessas matérias. Entende-se que para produzir os melhores resultados no processo de aprendizagem nessas áreas, faz-se constante a necessidade de atualização das didáticas de ensino de forma geral.

Face aos aspectos levantados, pretende-se com este estudo apresentar a importância da fundamentação de algoritmos e programação para alunos iniciantes em cursos de graduação na área da Computação, listando problemas de ensino nos conteúdos, além de algumas estratégias pesquisadas na literatura para tentar mitigá-los; e assim, pretende-se também possibilitar maior disseminação do conhecimento e promover a interatividade entre docentes e graduandos da área, enriquecendo a pesquisa com críticas e sugestões.

O artigo está organizado da seguinte forma: a seção 2 apresenta a definição e importância dos algoritmos na programação de computadores; a seção 3 discorre sobre alguns dos problemas acerca do ensino de programação; a seção 4 retrata a importância do uso de ferramentas computacionais no processo de ensino e aprendizagem de programação, e detalha algumas das ferramentas disponíveis; a seção 5, tece as considerações finais sobre o trabalho desenvolvido e direções para pesquisas futuras.

2. Conceitos e Alternativas para o Processo de Ensino-Aprendizagem de Algoritmos e Programação

Os algoritmos fazem parte do dia-a-dia de qualquer pessoa. As instruções para o uso de medicamentos e uma receita de culinária são exemplos de algoritmos. Um algoritmo pode ser entendido como uma sequência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema. Dijkstra (1971) afirma em seu livro "*A Short Introduction to the Art of Programming*", que um algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações. Ao executarmos a operação a + b percebemos um padrão de comportamento, mesmo que a operação seja realizada para valores diferentes de a e b.

Para resolver um problema é necessário escolher uma abstração da realidade, em geral através da definição de um conjunto de dados que representa a situação real. A seguir deve ser escolhida a forma de representar estes dados. A essa representação se chama: Estrutura de Dados (ED). Ou seja, algoritmos e ED estão intimamente ligados. Não se pode estudar ED sem considerar os algoritmos associados a elas, assim como a escolha dos algoritmos em geral depende da representação e da estrutura dos dados. Com essas definições, pode-se afirmar que Programar consiste em estruturar dados e construir algoritmos. De acordo com Rapkiewicz et al. (2006), programas são formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados. Onde programas representam uma classe especial de algoritmos capazes de serem seguidos por computadores.

No entanto, é importante saber que um computador só é capaz de seguir programas em linguagem de máquina, que correspondem a uma sequência de instruções obscuras e desconfortáveis; já uma linguagem de programação é uma técnica de notação para programar, com a intenção de servir de veículo tanto para a expressão do raciocínio algorítmico quanto para a execução automática de um algoritmo por um computador.

Não detendo-se aos detalhes das diferentes linguagens de programação, é possível perceber que a programação é um dos pontos chaves em um curso da área da computação. O seu entendimento e uso adequado de conceitos e tecnologias podem

conduzir ao desenvolvimento de produtos de *software* de qualidade. Com esse propósito, Setubal (2000) apresenta algumas recomendações gerais que podem ser aplicadas no ensino de disciplinas relacionadas ao ensino de algoritmos e de programação:

a. *Coerência com os objetivos fundamentais*: Nesse caso, o professor deve: i) expressar nitidamente as ideias, os conceitos e as técnicas aos alunos; ii) destacar a importância dos resultados teóricos e mostrar rigor formal toda vez que isto se fizer necessário; e iii) procurar valorizar o uso de técnicas na resolução de problemas. Esta última coerência pode ser alcançada em particular usando a técnica de descobrir a solução de um problema junto com os alunos, ao invés de simplesmente apresentar e explicar soluções prontas;

b. *Ênfase no pensamento crítico*: O professor deve manter atenção especial nesse quesito, dada a natureza teórica com forte componente matemático dessa disciplina/matéria. Os estudantes apresentam pouca maturidade matemática e tendem a acreditar em qualquer demonstração que lhes é apresentada. Tal comportamento deve ser desestimulado. É essencial que eles questionem aquilo que é apresentado, tendo em vista que é com dúvidas saudáveis e inquietação na resolução dos problemas que, a percepção da importância do resultado teórico, poderá ser consolidada. Assim, considera-se um recurso valioso o conjunto de exercícios que instigam os alunos a identificarem falhas de argumentação (por exemplo, erros em algoritmos para serem identificados);

c. *Teoria na prática*: a experiência com o ensino dessas disciplinas de Computação mostra que os alunos, em geral, não se sentem atraídos por elas, por considerarem-nas muito abstratas. Por esse motivo, crê-se ser importante usar como recurso didático, sempre que possível, um grande número de exemplos do cotidiano. A inclusão de projetos de implementação, ora dentro das disciplinas teóricas, ora dentro de uma disciplina específica, também visam tornar o ensino de programação menos abstrata.

Em geral, é importante salientar para os alunos iniciantes em cursos de computação o grande impacto que os resultados teóricos têm alcançado na prática, incentivando e estimulando para que os mesmos alcancem resultados satisfatórios.

3. Obstáculos no Ensino e na Aprendizagem de Algoritmos e Programação em Cursos de Computação

Segundo aponta Dos Santos (2005), disciplinas relacionadas à programação formam a essência dos cursos voltados para a área da computação, e o processo de aprendizagem de algoritmos é importante para a maioria das carreiras na área. Atualmente, o ensino de algoritmos e programação busca nas Ciências Exatas sua sustentação, pois despertam o raciocínio lógico e matemático para resolução de problemas.

A literatura apresenta algumas justificativas para a dificuldade ao aprender a programar: Autores como Dijkstra (1971), Almeida et al. (2002) e Rocha et al.(2010) argumentam que este tipo de aprendizagem deriva de um processo lento e gradual ou falta de interesse por parte dos alunos. Referem ainda que há uma desmotivação para estas atividades, e que está relacionada a uma forte carga de conceitos abstratos que intervêm em todo o conhecimento envolvido em programar, onde as características próprias das linguagens e ambiente de programação, cada vez mais sofisticados, e da máquina em si, tendem a dificultar. Adicionalmente, Rocha et al.(2010) afirmam que a maioria das linguagens de programação utilizadas nas disciplinas introdutórias apresenta uma sintaxe grande e complexa, mais adequada para ambientes de

desenvolvimento industrial.

Outros autores contribuem para essa análise, como Pereira e Rapkiewicz (2004), dizendo que mesmo antes do começo das atividades já se forma uma resistência perante ao conteúdo, fato que pode ocorrer por sua parte lógica e matemática envolvidas, dado o aprofundado estigma incluído nesse conteúdo. Raabe e Silva (2005) avaliaram algumas dificuldades existentes nessa área de ensino e as classificam em três grandes dimensões que podem dar origem ao problema, são elas: a didática, a cognitiva e a afetiva. Para os autores, problemas que abrangem o ritmo de aprendizado dos alunos, materiais e métodos utilizados, o perfil comportamental ou mesmo problemas pessoais e afetivos são apontados como relevantes. Por conta destes problemas, Borges (2000) diz que o índice de alunos que completam seu curso sem ter um conhecimento mínimo adequado na área de programação é alto. Ou ainda, muitos não completam o curso, abandonando-o nos primeiros semestres devido às dificuldades.

Para Tobar et al. (2001), outro fator agravante é a dificuldade encontrada pelos professores em acompanhar efetivamente as atividades práticas em laboratório de programação, dado o grande número de estudantes sob sua supervisão. Além disso, ocorrem situações em que o aluno, uma vez que consegue resolver um determinado problema, não se interessa em verificar se é a solução mais simples ou adequada, satisfazendo-se com os resultados iniciais obtidos (Borges, 2000; Giraffa et al., 2003). Muitas vezes, o reduzido tempo para atividades de laboratório também contribui para que isto ocorra (Rocha et al., 2010).

4. Ferramentas e Estratégias para o Ensino de Programação

Para Menezes e Nobre (2002), Júnior et al. (2005), Soares et al. (2004) entre outros, afirmam que os problemas apresentados na compreensão de conceitos abstratos de programação encontra sua raiz ainda durante a formação básica, o que caracteriza uma importante lacuna a ser pesquisada.

Dessa forma, o acompanhamento e o tratamento diferenciado a cada aluno, principalmente em turma com um número elevado de alunos, serão possíveis se forem apoiados por sistemas inteligentes, ou seja, auxiliados por computador (Pimentel et al., 2003). Dessa forma, a tentativa de usar *softwares* para o ensino de programação passa a ter grande importância, e ainda poderá transformar o ensino de disciplinas avançadas em algo mais primoroso e tecnológico (Dos Santos, 2005).

Juntamente com o uso de ferramentas computacionais, deve-se tentar ajustar algumas estratégias de trabalho que tem atraído a atenção e o desenvolvimento de muitas pesquisas na área (Júnior e Rapkiewicz, 2004). Tentando solucionar parte das dificuldades relacionadas ao ensino de programação, diversos trabalhos têm sido desenvolvidos no âmbito acadêmico, exemplos desses estudos que apresentam algumas dessas ferramentas, e as suas vantagens a área, estão destacados na Quadro 1.

Quadro 3 - Ferramentas para o Ensino de Algoritmos e Programação

Nome	Descrição	Análise
ASTRAL *Animation of Data Structures and Algorithms* (Garcia, Rezende e Calheiros, 1997)	Ambiente de programação para produção de animações de algoritmos e estruturas de dados. Implementa uma variedade de estruturas de dados e algoritmos e, através de chama\das de algumas rotinas específicas do ambiente, é possível visualizar graficamente as estruturas de dados.	Aprendizado beneficiado pela observação dinâmica do funcionamento das estruturas de dados; Proporciona um nível de abstração maior, e depuração, que é facilitada pela visualização gráfica, deixando claro ao programador, no momento da execução, em que parte da implementação da estrutura ou do algoritmo os erros ocorrem.

TED Tutorial de Estruturas de Dados. (Flávio, 2004)	Tutorial aplicado no ensino de estrutura de dados. Permite uma maior interação entre o professor e o aluno, servindo como uma ferramenta de apoio ao aprendizado.	Através de conceitos de dinamização de aprendizagem, como a visualização dos acontecimentos simultaneamente com a execução, o aluno cria uma visão de totalidade do sistema.
***DICTIONARY OF ALGORITHMS AND DATA STRUCTURES*.** (Black, 2005)	É um site que contém parte do *Software Quality Group do Software Diagnostics and Conformance Testing Division, Information Technology Laboratory*, que contém um dicionário de algoritmos, estruturas de dados, problemas atípicos e definições relativas.	Alguns dos algoritmos disponíveis no dicionário têm links para implementações em linguagem de programação C e C++. Os códigos são comentados e permitem ao aluno perceber os detalhes da implementação apresentada.
DIDAGRAPH (Dagdilelis e Stratzemi, 1998)	*Software* para o ensino de algoritmo através da teoria dos grafos. Permite acompanhar animações da execução de algoritmos por meio de uma visualização do grafo com o algoritmo em andamento e de uma descrição em linguagem de alto nível do algoritmo.	Apresenta poucos algoritmos. É necessário aprender uma nova linguagem, desviando do objetivo das disciplinas de grafos; Possui alguns problemas: (i) ausência de textos explicativos sobre o status de execução; (ii) ausência de metáforas para representar botões e recursos de interação; e (iii) execução contínua (não permite paralisar o processo).
EVEGA (Khuri e Holzapfel, 2001)	É uma ferramenta desenvolvida para o aprendizado de algoritmos em grafos e fornece uma interface poderosa e de boa usabilidade para criação e edição interativa de grafos e animação de algoritmos.	As execuções podem ser comparadas por meio de gráficos de desempenho gerados pela ferramenta. Um algoritmo de fluxo máximo acompanha a ferramenta para demonstração e é possível incluir no ambiente novos algoritmos. A operação para geração das animações é feita de forma intrusiva no código do algoritmo, podendo ser necessário dividir a atenção do problema a ser solucionado com a criação da animação.
VisualGraph **(**Lucas e Naps, 2003).	Biblioteca para a animação de algoritmos em grafos que não fornece uma interface gráfica para o usuário, servindo como base para a criação de grafos que possam ser utilizados em outros projetos e ambientes.	Para visualizar a animação, é necessário o uso de outra ferramenta. A interface para criação de grafos é baseada em texto, pouco intuitiva ao aluno;
DisViz (Sherstov, 2003)	Foi desenvolvido para permitir o aprendizado em grupo, por meio de redes locais no qual é possível criar grafos de teste e animações de algoritmos que possam ser visualizados por todos os membros da rede	Não fornece uma maneira fácil de acoplar algoritmos desenvolvidos; os alunos só podem utilizar os algoritmos existentes. Também depende da existência de uma rede local e da colaboração dos membros dessa rede, o que pode dificultar o estudo dentro e fora dos horários de aulas práticas.
GVF – Graph Visualization Framework (Marshall e Herman, 2003)	É um framework para desenvolvimento de ferramentas direcionadas para manipulação de problemas em grafos que apresenta um bom suporte para criação de entrada e saída de dados e layout para os grafos.	Por outro lado, não permite a inclusão de algoritmos dinamicamente ao sistema e não fornece amparo à animação de algoritmos. Não fornece capacidade de expansão, pois existe a possibilidade de inclusão de operações, componentes e propriedades de um grafo.
TBC-AED e TBC-AED GRAFOS Treinamento Baseado em Computador para Algoritmos e Estruturas de Dados e em Grafos Grafos. (Dos Santos e Costa, 2005)	Foram desenvolvidas visando aumentar a agilidade em aulas teóricas ao melhorar a visualização da execução de algoritmos e propiciar espaço às atividades práticas. Ambas tiveram sua versão adaptada para web, gerando TBC-AED/WEB e TBC-GRAFOS/WEB, a fim de evitar a necessidade de download das ferramentas.	Apresenta interface que facilita a aprendizagem, linguagem simples; botões habilitados, a cada momento, para o uso adequado das ferramentas; links explicativos; processo gráfico passo a passo com elementos numéricos, melhorando a visualização e o entendimento; legendas explicativas, ilustrando etapas de execução dos algoritmos; e conteúdo teórico simples.
EDDL Estruturas de Dados Dinâmicas Lineares (Azul e Mendes, 1998)	Ambiente computacional que engloba: uma abordagem introdutória aos conceitos ou especificação dos tipos de estruturas de dados; as técnicas de implementação; e a utilização dessas estruturas na resolução de problemas. Disponível para o ambiente Windows.	Oferece ferramentas de programação que o tornam intuitivo, de fácil manipulação, diversificado, versátil e eficaz para o desenvolvimento; Privilegia o aspecto gráfico, para visualização do fluxo de execução e erros do algoritmo; Conteúdos que podem ser trabalhados: listas lineares, filas, pilhas.

Como é possível observar, há diversas ferramentas construídas para o ensino de algoritmos e programação. Cada trabalho apresenta um foco relacionado às necessidades de seus desenvolvedores e professores interessados. Analisando esses trabalhos, é possível observar que as ferramentas, na maioria, trabalham com ambientes gráficos e de execução passo a passo para auxílio ao ensino e aprendizagem, e tem como base a linguagem de programação Java e C++. Os resultados demonstram que as ferramentas gráficas facilitaram a visualização das abstrações dos algoritmos implementados. A figura 1, por exemplo, mostra um algoritmo de Árvore Binária de Busca implementado no *software* TBC-GRAFOS (Dos Santos e Costa, 2005).

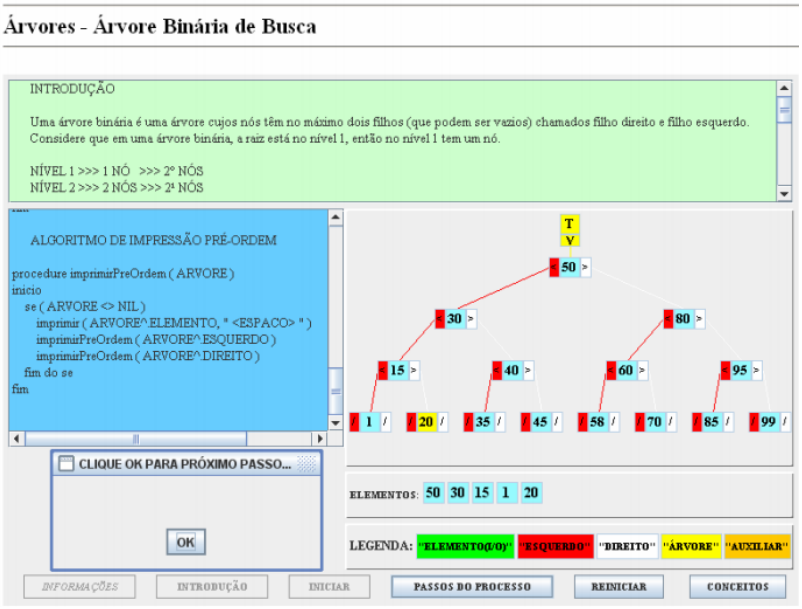

Figura 13 - Tópico Árvore Binária de Busca.

Fonte: Dos Santos e Costa, 2005

Nos estudos de Dos Santos e Costa (2005), Marshall e Herman (2003), Garcia et al. (1997), Dagdilelis e Stratzemi (1998), Khuri e Holzapfel (2001), Lucas e Naps (2003), Sherstov (2003) os resultados apontam para redução no índice de reprovação da disciplina programação devido ao emprego desses *softwares* que apresentam como principal característica a usabilidade e autonomia, por serem auto explicativos. Onde o estudantes só precisam deslizar com o *mouse* nas partes da tela para ver uma breve descrição sobre a região posicionada. Isso deixa o utilizador à vontade e sem preocupações quanto a quaisquer peculiaridades, também por se tratar de um produto de *software* direcionado para alunos iniciantes em cursos de Computação.

Dentre as vantagens da abordagem construtiva definida por Garcia et al. (1997), estão: mecanismos para facilitar o processo de abstração, o fato da animação refletir a interação com o aprendiz e as várias facilidades para a detecção visual de erros. Com isso, incentiva-se o processo de compreensão e autocorreção. Isso pode ser facilmente percebido em processos recursivos, os quais são de difícil explicação teórica, mas que podem ser vistos facilmente por meio da animação gráfica disponível em ambientes que utilizam algum algoritmo desse tipo (como *merge sort e quick sort*). Com isso, verifica-se que a organização de ferramentas gráficas é amplamente didática e essa característica é muito útil para o ensino de disciplinas voltadas para programação.

Além dessas possibilidades, é uma experiência desafiadora para alunos e educadores em tornar o ensino de programação mais dinâmico, ao despertar o seu

interesse a terem a mesma iniciativa, se possível com igual dedicação e empenho, para melhorar e aprimorar a formação de recursos humanos para a tecnologia. Das ferramentas pesquisadas, apenas o *Dictionary of Algorithms and Data Structures*, proposta por Black (2005), o autor não se preocupa com o aspecto visual da sua implementação da ferramenta, nesse sentido, não foi possível localizar trabalhos com resultados concretos a partir deste.

5. Considerações Finais

A respeito do conjunto de informações obtidas, pode-se perceber que existe uma gama de recursos que possibilitam o desenvolvimento de ferramentas computacionais para o auxílio do ensino de algoritmos e programação, com destaque para as linguagens Java e C++, com predomínio da linguagem Java em relação ao C++. Observou-se também, que a opção por usar a visualização gráfica do comportamento do algoritmo é unânime nos casos pesquisados, onde, não ocorre uma exploração direta de código fonte, minimizando o aspecto sintático e semântico do algoritmo, mas sim o caminho percorrido por ele até a solução desejada.

Verificou-se também que uso de *software* para o ensino de programação e algoritmos é uma ideia interessante, se for amadurecida e desenvolvida de forma cuidadosa e estruturada. Isso fornece novas experiências para professores e alunos, que se proponham a trabalhar com esse tema. Além disso, eles poderão melhorar a qualidade de ensino e ser capazes de avançar mais rapidamente no campo do conhecimento, através da melhoria dos processos didáticos e da participação ativa do docente da área.

Conclui-se também que uma boa elaboração de produtos tecnológicos que facilitem a transmissão de conhecimentos em Computação deve ser acompanhada de expressiva pesquisa no campo de novas metodologias de ensino. A principal finalidade disso é cada vez mais haver contribuições que incorram no aprimoramento do ensino superior e na formação de profissionais melhor qualificados para o mercado. Isso reflete principalmente sobre futuros professores, uma vez que aqueles que têm passado por esse tipo de experiência durante a graduação terão grande interesse em executar (e explorar) esse processo quando estiverem lecionando. Nesse sentido, como trabalhos futuros pretende-se executar experimentos (quantitativos) com diferentes abordagens e coletar métricas de cada uma das técnicas (ferramentas) analisadas.

Referências

ALMEIDA, E. S. et al. (2002). AMBAP: Um Ambiente de Apoio ao Aprendizado de Programação. Anais do X Workshop sobre Educação em Computação, Florianópolis. Brasil.

ASTRAL (2016). Disp. em: < http://bit.do/cPuZ3 > Acesso em 18 abr. 2016.

AZUL, A. A.; MENDES, A. J. (1998) EDDL: Um Programa Didático sobre Estruturas de Dados Dinâmicas Lineares. Anais do III Simpósio Investigação e Desenvolvimento de Software Educativo. Évora, Portugal.

BLACK, P. E. (2005) Dictionary of algorithms and data structures. NIST - Institute of Standards and Technology.

BORGES, M. A. F. (2000) "Avaliação de uma Metodologia Alternativa para a Aprendizagem de Programação". VIII Workshop de Educação em Computação. Curitiba, PR, Brasil.

CHARLTON, P.; LUCKIN, R. (2012) Time to reload? Computational Thinking and Computer Science in Schools. What researches says? Briefing 2. London Knowledge Lab - Institute of Education, London, UK.

DAGDILELIS, V.; STRATZEMI, M. (1998) DIDAGRAPH: Software for Teaching Graph Theory Algoritms. ITiCSE 1998, Integrating Technology into Computer Science Education. Dublin, Irlanda.

DIJKSTRA, E.W. (1971) A Short Introduction to the Art of Programming. Technological University Endhoven, Países Baixos.

DOS SANTOS, A. et al. (2005). TBC-AED: Um Software Gráfico para Apresentação de Algoritmos e Estruturas de Dados aos Iniciantes em Computação e Informática. Disp. em: < http://bit.do/cPsRT > Acesso em 13 abr. 2016.

FLÁVIO, D. (2004) TED: Tutorial de Estruturas de Dados, desenvolvido durante estágio supervisionado na Universidade do Vale do Itajaí. Disp. em < http://bit.do/cPsud > Acesso em 20 jun 2016.

GARCIA, I. C. et al. (1997). Astral: Um Ambiente para Ensino de Estruturas de Dados através de Animações de Algoritmos. Revista Brasileira de Informática na Educação, Florianópolis, SC, v. 1, p. 71-80.

GIRAFFA, L. (2003) O Ensino de Algoritmos e Programação mediado por uma Ambiente Web. Anais do Congresso Nacional da Sociedade Brasileira de Computação. Campinas, Brasil.

JÚNIOR, J. C. R. P. e RAPKIEWICZ, C. E. (2004). O Processo de Ensino e Aprendizagem de Algoritmos e Programação: Uma Visão Crítica da Literatura. Anais do III Workshop de Educação em Computação e Informática do estado de Minas Gerais. Belo Horizonte, MG, Brasil.

JÚNIOR, J. C. R. P. (2005). Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio. Anais do XIII Workshop de Educação em Computação. São Leopoldo, RS, Brasil.

KHURI, S.; HOLZAPFEL, K. (2001). An Education Visualization Environment for Graph Algoritms. Proceedings of ITiCSE 2001, The 6th Annual Conference on Innovation and Technology in Computer Sciene Education. Canterbury, UK.

KULYK, O. et al (2007). Human-centered visualization environments. Lecture Notes in Computer Science, v. 4417, p. 13-75.

LUCAS, J. M. e NAPS, T. L. (2003) VisualGraph – A Graph Class Designed for Both Undergraduate Students and Educators. SIGCSE 2003, Special Interest Group on Computer Science Education. Reno, Nevada, USA.

MARSHALL, G. M. M. S.; HERMAN, I. (2003). An Object-Oriented Design for Graph Visualization.

MENEZES, C. e NOBRE, I. (2002) Suporte à Cooperação em um Ambiente de Aprendizagem para Programação (SambA). Anais do XIII Simpósio Brasileiro de Informática na Educação. São Leopoldo, RS, Brasil.

PEREIRA, J.C.R.; RAPKIEWICZ, C. (2004) O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma Visão Crítica da Pesquisa no Brasil. Workshops de Educação em Informática.

PIMENTEL, E. P. et al (2003). A Caminho de um Ambiente de Avaliação e Acompanhamento Contínuo de Aprendizagem em Programação de Computadores. Anais do II Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG 2003). Poços de Caldas, MG, Brasil.

RAABE, A. L. A.; SILVA, J. M. C (2005). Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. XXV Congresso da Sociedade Brasileira de Computação. São Leopoldo/RS.

RAMOS, J.L.; ESPADEIRO, R.G. (2014) Introducing computational thinking in pre-service teacher education. In: Atas 3º Congreso Ibero-Americano de Investigation Cualitativa.

RAPKIEWICZ, C. E. et al. (2006). Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. Ciclo de Palestras Novas Tecnologias na Educação - UFRGS, Porto Alegre.

ROCHA, P. S. et al. (2010). Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. Novas Tecnologias na Educação. CINTED-UFRGS, v. 8, n. 3, 2010. Disp. em: < http://bit.do/cPsA2 > Acesso em 10 mar. 2016.

SETUBAL, J. C. (2000) Uma proposta de Plano Pedagógico para a Matéria de Computação e Algoritmos. Anais do II Curso: Qualidade de Cursos de Graduação da Área de Computação e Informática (WEI). Editora Universitária Champagnat.

SHERSTOV, A. A. (2003). Distributed Visualization of Graph Algoritms. Special Interest Group on Computer Science Education. Reno, Nevada, USA.

SOARES, T.C.A.P. et al. (2004) Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não-Intrusiva de Algoritmos. Anais do III Workshop de Educação em Computação e Informática de Minas Gerais.

TOBAR, C. M. et al (2001) R. Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação. Anais do XII Simpósio Brasileiro de Informática na Educação. Vitória, ES, Brasil.