Uma Oficina para Ensino de Algoritmos Paralelos por Meio de Computação Desplugada

Anderson C. Lima[1], **Daniel Bailo** [1], **Thiago Carvalho** [1], **José Filipe R. Rodrigues** [1], **Plabiany R. Acosta** [1], **Wellington M. Aquino** [1]

[1]Universidade Federal de Mato Grosso do Sul
Campus de Ponta Porã (CPPP) - BR 463 - Km 4,5 - CEP 79907-414
Ponta Porã, Brasil

Abstract. *The parallel programming is already present in the equipment of the most diverse sciences. Parallel computing made it possible to solve problems that were impractical in sequential programming. However, learning the particularities of computational thinking in parallel is not a simple task. It is necessary to recognize the urgency of the teaching of this discipline in courses of technical schools or undergraduate informatics in Brazil. This paper describes an account of classroom experience through workshops, with the use of unplugged computing to teach some of the first concepts of parallel algorithms.The test applied at the end of the workshops presented satisfactory results.*

Resumo. *A programação paralela já esta presente em equipamentos das mais diversas ciências. A Computação Paralela tornou possível a solução de problemas que eram impraticáveis em programação sequencial. Entretanto, aprender as particularidades do pensamento computacional em paralelo não é uma tarefa simples. É preciso reconhecer a urgência do ensino desta disciplina em cursos de escolas técnicas ou de graduação em informática no Brasil. Este trabalho descreve um relato de experiência em sala de aula através de oficinas, com uso de computação desplugada para ensinar alguns dos primeiros conceitos de algoritmos paralelos. O teste aplicado ao final das oficinas apresentou resultados bastante satisfatórios.*

1. Introdução

Habitualmente, as escolas de ensino básico propiciam o contato com a informática por meio da exposição de alguns conceitos iniciais e enfatizam atividades práticas, as quais requerem uso dos computadores em laboratórios de informática. Entretanto, levando em consideração o desenvolvimento acelerado dos aparatos tecnológicos no mundo contemporâneo e o fato de os temas pesquisados não serem inéditos, tendo em vista as pesquisas de Papert realizadas há mais de 30 anos [Papert 1980], as ações inovadoras das escolas tem sido pontuais [Lessa et al. 2015]. Já no ensino superior ou médio técnico nas áreas de informática, a proposta curricular é geralmente direcionada para o ensino de habilidades com foco na tradicional programação sequencial [Vieira et al. 2013].

No entanto, o ensino apenas de programação sequencial não condiz mais com a realidade dos níveis tecnológicos existentes na computação hoje, principalmente após 2003, quando surgiu um novo paradigma para computação que se baseou na utilização de mais

DOI: 10.5753/cbie.wcbie.2018.619

de uma unidade de processamento, conhecidas como núcleos (*cores*), em cada chip, com o propósito de aumentar o poder de processamento [Kirk and Hwu 2010]. Era o início da computação paralela de alto desempenho, que catapultou a capacidade de processamento de diversos dispositivos tecnológicos, muitos deles úbiquos [Kirk and Hwu 2010].

A Computação Paralela e Distribuída (*Parallel and Distributive Computing* - (PDC)) de Alto Desempenho já faz parte do dia-a-dia de muitas pessoas, mesmo que elas não saibam. Ela está presente no processamento de seus *smartphones*, *websites* de redes sociais e muito mais. Tornou-se urgente então transmitir e ensinar as particularidades do pensamento computacional paralelo em vários níveis da estrutura educacional [Prasad et al. 2013]. Isto é primordial e imperativo, visto que o advento da computação paralela de alto desempenho parece não admitir retrocesso. Entretanto, ensinar o pensamento computacional "em paralelo", principalmente para novos programadores, não é uma tarefa simples. Muitas habilidades precisam ser treinadas. Na educação em informática, seja ela em nível básico, técnico, tecnológico ou superior, este parece ser um grande desafio. Certamente, não é mais suficiente que os futuros programadores adquiram somente habilidades da programação sequencial convencional[Prasad et al. 2011].

É possível encontrar algumas iniciativas atuais fora do Brasil, que procuram incentivar o ensino da computação paralela na sociedade acadêmica e tecnológica, do ensino médio ao superior em informática [Torbert et al. 2010, Gregg et al. 2012, Bogaerts 2017, Saule 2018]. Entretanto, no Brasil, iniciativas com novas práticas para o ensino de computação pararela parecem ainda ser pouco discutidas.

O ensino da programação paralela é uma tarefa complexa, pois, diferentemente da programação sequencial, ela possui uma série de etapas e diferentes modelos de programação. Entretanto, a procura por abordagens mais pedagógicas para o ensino desta disciplina já vem se ampliando dentro de instituições de ensino estrangeiras. Consideramos que no Brasil ainda é preciso despertar o interesse dos novos programadores por esta área e rever as metodologias das instituições de ensino para a computação paralela.

Este trabalho propõe uma metodologia diferente para ensinar computação paralela. Para desmistificar os conceitos iniciais, que podem parecer de difícil aprendizado, se utilizou práticas advindas da computação desplugada, ou seja, sem o uso de computador. Para tal, foi desenvolvido uma maquete real construída partir de garrafas PET para ensinar os conceitos fundamentais dos algoritmos paralelos. Ainda neste trabalho é descrito oficinas realizadas em aulas presenciais em duas instituições de ensino Federais: 1) A primeira de ensino Técnico Federal do curso integrado em informática (IFMS - Dourados). 2) A segunda de Graduação em Computação (UFMS - Campus Ponta Porã). Nesta experiência foi utilizada a maquete construída para interagir com os estudantes durante a aula. As oficinas foram realizadas dentro da disciplina de Algoritmos I em ambas as instituições. Um teste foi proposto aos estudantes em função de verificar o poder didático do método em questão. Os teste apresentaram um índice de acerto de cerca de oitenta por cento, e tais resultados apontam uma eficiência representativa do modelo proposto.

2. Fundamentação Teórica

São muitas as iniciativas internacionais que reconhecem a necessidade da atualização dos seus sistemas educacionais no que se refere à educação em computação. Na literatura diversos trabalhos evidenciam a necessidade de ampliar os esforços dos pesquisadores no

que se refere ao conhecimento que se possui sobre o porquê de ensinar computação nas escolas, de como os conteúdos deveriam ser ensinados, que tópicos deveriam ser lecionados e para quem esta educação poderia ser significativa [Cambraia and Scaico 2013]. Particularmente, estes temas ainda geram muitas discussões, principalmente quando consideramos a constante evolução tecnológica e seus reflexos no ensino de computação.

As disciplinas de computação são compostas de diferentes e interligadas áreas de conhecimento desenvolvendo diferentes competências, habilidades e conhecimentos tanto na área básica quanto na área tecnológica. Entre as competências mais difíceis de serem desenvolvidas estão as relacionadas com o desenvolvimento de algoritmos e programas [Júnior et al. 2005]. Várias são as dificuldades pelas quais passam os alunos durante o processo de ensino aprendizagem de programação. A modularização do programa ou a retirada de erros e conceitos mais abstratos, como recursão e ponteiros são algumas das dificuldades citadas. Contudo, a maior delas está relacionada à combinação e à utilização apropriada dos conceitos básicos de programação para a construção de um programa. Desta forma, os alunos parecem entender os conceitos e as estruturas que compõem uma linguagem de programação, mas não sabem como utilizá-las corretamente durante a construção de seus próprios programas [Aureliano and Tedesco 2012]. Foi notada, então, uma forte razão para incluir tal temática desde cedo na educação, pois assim, além de outros benefícios, os futuros alunos dos cursos de computação teriam menos dificuldade de aprendizado. E aqueles que se dirigissem para outros ramos profissionais teriam desenvolvido competências de resolução de problemas e formalização dos mesmos que seriam úteis em suas respectivas áreas do conhecimento [Júnior et al. 2005].

2.1. O computador paralelo e seu primeiro modelo

Segundo Ian Foster, um computador paralelo é um conjunto de processadores interconectados, capazes de trabalhar de forma simultânea e cooperativa a fim de resolver um mesmo problema computacional [Foster 1995]. Essa definição é suficientemente ampla para incluir supercomputadores paralelos, que englobam centenas ou milhares de computadores, redes locais de computadores e estações de trabalho com múltiplos processadores. A ideia principal da computação paralela consiste em executar tarefas simultaneamente. Uma tarefa pode realizar quatro ações básicas, além de ler e escrever em sua memória local: enviar mensagens, receber mensagens, criar novas tarefas e terminar. Canais podem ser criados e excluídos, e as referências aos canais (portas) podem ser incluídos em mensagens, de modo que a conectividade pode variar dinamicamente [Foster 1995]. Percebe-se então que os conceitos de computação paralela são bastante específicos e necessitam de muita abstração para uma correta compreensão de seus significados, principalmente para os programadores iniciantes. Em 1979 o modelo PRAM de James C. Wyllie [Wyllie 1979], foi o primeiro modelo teórico proposto para a compreensão de alguns conceitos introdutórios de algoritmos paralelos.

Na última década, a criação de microprocessadores com múltiplos núcleos tornou-se uma tendência, principalmente quando a indústria percebeu que o aumento da frequência de processamento estava criando mais problemas do que resolvendo, devidos entre outros fatores aos problemas de superaquecimento dos chips com poucos *cores* [Kirk and Hwu 2010]. Seguindo esta tendência computadores em um contexto geral passaram a suportar múltiplas *threads* (*multithreading*) ou múltiplos núcleos (multi/*many core*). Hoje em dia, os *hardwares* paralelos compostos de múltiplos processadores, já

estão presentes nos principais equipamentos ou plataformas computacionais. Estações de trabalho, servidores, supercomputadores ou até mesmo sistemas embarcados já fazem uso desta arquitetura [Kirk and Hwu 2010]. A presença de mais processadores aumentou drasticamente o poder computacional de diversos dispositivos e isto possibilitou o tratamento de problemas, que há bem pouco tempo eram impraticáveis. Novas abordagens paralelas, proporcionadas pelas novas arquiteturas de múltiplos núcleos, ofereceram então novas propostas para solução de antigos problemas. O desafio de explorar estas novas abordagens sinaliza ser bastante interessante, dada a possibilidade de ganho de tempo, porém, antes de qualquer jornada neste caminho, é necessário compreender e utilizar as novas arquiteturas de multiprocessamento da melhor maneira possível. Os obstáculos se iniciam precocemente, visto que pensar em paralelo, por si só exige maior esforço do programador. É preciso tratar de problemas de concorrência e sincronização, para os quais, a depuração é um processo bastante custoso e as vezes manual. Logo, percebe-se que mesmo com as arquiteturas multi/*many core* sendo uma realidade, a programação paralela continua a ser uma tarefa complexa.

Além da dependência e disponibilidade de ferramentas e ambientes de programação adequados para memória compartilhada ou distribuída, enfrenta-se uma série de problemas não existentes em programação sequencial: código paralelo, concorrência, comunicação, sincronização, granularidade e balanceamento de carga, estão entre eles [Kirk and Hwu 2010]. Como visto, são diversos os obstáculos, mas se não forem criados programas que façam uso dos recursos da computação paralela multi/*many core*, o desperdício de processamento e de energia continuará [Kirk and Hwu 2010], entretanto uma outra questão ainda mais relevante é: ***Visto que a computação paralela de alto desempenho é bastante complexa, mas tornou-se imprescindível, qual a melhor forma de ensinar seus conceitos e técnicas aos futuros programadores?***. A resposta para esta pergunta não é simples, pois desenvolver *software* baseado em computação paralela é uma tarefa de programação árdua e de considerável esforço intelectual. Além disso, adquirir dispositivos com milhares de núcleos para o ensino de programação paralela é financeiramente custoso para as intituições públicas de ensino.

2.2. O Ensino da Programação Paralela: Breve Revisão da Literatura

Iniciativas sobre novos métodos a respeito do ensino de computação paralela já vem sendo discutidas em diversos trabalhos internacionais. Em um trabalho recente, de 2017, compilou-se, por meio de uma revisão sistemática, o estado da arte sobre os principais métodos para o ensino de computação paralela disponíveis [Bachiega et al. 2017]. A maior parte deles envolve práticas com a utilização de *softwares* e computadores. O mesmo trabalho apresenta a utilização de sala de aula invertida como uma prática, destacando que um ponto relevante é desenvolver metodologias criativas que se adaptem na estrutura que faz parte da realidade dos alunos e das condições financeiras de suas escolas. Em algumas instituições de ensino, por exemplo, há apenas um microcomputador para um ou mais estudantes. Algumas das propostas descritas naquele trabalho, que também estão presentes em outros artigos são apresentadas a seguir: 1) Um exemplo dessas iniciativas foi a proposta apresentada pelo Comitê Técnico do IEEE sobre Processamento Paralelo (TCPP), junto com a *National Science Foundation* (NSF) [Prasad et al. 2011]. A proposta descreve uma nova estrutura curricular para os cursos de Ciência da Computação e Engenharia da Computação, de forma que os graduandos possam adquirir um certo nível

de habilidades no estudo de computação paralela. Aprender as técnicas e tecnologias da computação paralela tem se tornado um desafio tanto para os profissionais já na área de desenvolvimento, como também para os que agora se iniciam no estudo da computacão, seja ela no ensino técnico ou superior [Cesar 2017]. 2) Particularmente, um dos exemplos bem sucedidos de uma experiência do ensino do pensamento paralelo para alunos do nível médio aconteceu no formato de um teste piloto na escola Thomas Jefferson *High School for Science and Technology* [Torbert et al. 2010]. No método proposto os problemas evoluem ao longo do curso, de forma a gerar uma compreensão da amplitude da área da computação paralela ao mesmo tempo em que eles podiam exercitar a solução de problemas clássicos da computação paralela. O grande diferencial foi a utilização de um sistema com uma linguagem para programação paralela, que foi considerada por eles, como sendo de mais fácil aprendizado do que aquelas normalmente utilizadas. O sistema denominado XMT (*EXplicit Multi-Threading*) foi projetado pela universidade de Maryland e permitiu que alunos iniciantes aprendessem programação paralela partindo de problemas clássicos e conhecendo suas soluções em paralelo de forma natural. 3) Em um trabalho recente Saule e Erik [Saule 2018] destacam que mesmo no ensino superior inserir a computação paralela, nos primeiros anos de um currículo de bacharelado em cursos de informática é visto como um problema atual e difícil. Os autores ressaltam que em muitas instituições os docentes podem não ser capacitados no processo de um ensino pedagógico da disciplina e com isto o aprendizado pode se tornar mais difícil para os alunos. Os autores salientam que é um problema atual a necessidade de bons exemplos de como ensinar computação paralela. Em seu trabalho eles destacam que a disciplina de computação paralela é obrigatória nos cursos de informática da universidade UNC Charlotte nos Estados Unidos da América (EUA). Os autores apresentam um relato de experiência onde foi aplicada uma categorização para o ensino de computação paralela na turma de outono de 2017 de alunos matriculados na disciplina [Saule 2018]. Por fim, é importante destacar que das metodologias encontradas na literatura não encontrou-se nenhuma que fizesse uso de computação desplugada, prática proposta como parte da oficina que será apresentada neste trabalho.

2.3. A Computação desplugada

A expressão Computação Desplugada tornou-se conhecida após publicação do livro "*Computer Science Unplugged-off-line activities and games for all ages*", em 1998, dos autores Tim Bell, Ian H. Witten e Mike Fellows [Bell et al. 1998]. A Sociedade Brasileira da Computação (SBC) entende que os conceitos da Computação devam ser ensinados a partir do ensino básico e incentivam ações dessa natureza [Costa et al. 2012]. Dentre as iniciativas existentes pode-se citar o projeto da Computação Desplugada [Bell et al. 1998]. Este projeto apresenta alternativas de ensino da computação por meio de atividades lúdicas e analogias do cotidiano, não utilizando recursos de hardware e/ou software. Este método de trabalho tem despertado interesse de pesquisadores e professores em diversos países, sendo sua divulgação feita através do site (http://www.csunplugged.org/) e do livro de atividades que atualmente já possui a tradução para o português [Costa et al. 2012]. O livro apresenta uma compilação de várias atividades e dinâmicas que apresentam, de forma lúdica, conceitos matemáticos e computacionais como números binários, padrões e ordenamento, criptografia, dentre outros. A computação desplugada pode ser vista como uma democratização do ensino da computação, pois, algumas de suas vantagens abrangem a possibilidade de se ensinar a

todas as idades [Vieira et al. 2013]; com independência dos recursos de *hardware* e *software*, o que gera a possibilidade do ensino em áreas remotas e sem acesso a tecnologia; a viabilidade de ser aplicada por não especialistas na área; dentre outras [Bell et al. 2011].

3. Materiais e Métodos

Este trabalho tem como objetivo apresentar uma forma alternativa para o ensino dos conceitos introdutórios e iniciais de algoritmos paralelos. A metodologia aplicada foi baseada em técnicas de computação desplugada e na interação dos estudantes com uma maquete, que representa o fluxo de execução de um algoritmo específico. Por sua simplicidade, inicialmente, o primeiro algoritmo escolhido foi o algoritmo da soma de um vetor númerico em paralelo [JáJá 1992] no modelo teórico PRAM. Um exemplo do funcionamento deste algoritmo é representado pela Figura 1[Vishkin 1993]. Para cumprir o objetivo programado a metodologia foi dividida em três etapas: 1) Construção da maquete, que representa o algoritmo paralelo da soma de vetor, com garrafas PET. 2) Elaboração de uma apresentação expositiva e pedagógica, no formato de aula, denominada: "Computação Paralela: Despertando e Desplugando". 3) Realização de duas oficinas em turmas iniciantes da disciplina de Algoritmos I, em duas instituições Federais, uma de ensino técnico e outra de ensino superior. A maquete foi construída a partir de garrafas de PET grandes e médias, bolinhas de isopor, pranchas de isopor, tintas guache, fitas coloridas e outros itens mais acessíveis e com baixo custo. Como já dito, a maquete simula o fluxo de execução de um dos primeiros e principais algoritmos para o ensino inicial de computação paralela: o algoritmo da soma de um vetor numérico em paralelo, descrito em detalhes no livro de Joseph JáJá [JáJá 1992]. A Figura 2 apresenta a maquete que projetamos e construímos.

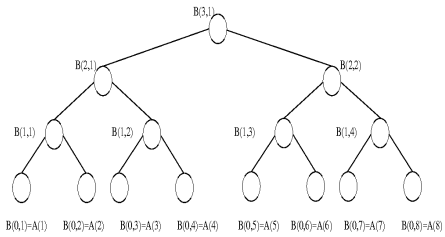


Figura 1. Algoritmo da soma em paralelo em modelo teórico [Vishkin 1993].

Figura 2. Maquete do algoritmo da soma em paralelo.

4. Resultados e Discussão

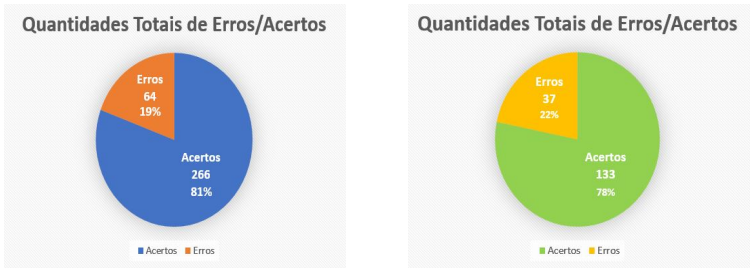
A oficina realizada nas duas instituições de ensino, a interação com os alunos e as percepções do experimento fazem parte de nossos resultados. Foi iniciada a oficina com uma apresentação no formato de aula, de cinquenta minutos, em *datashow* sobre a importância de se aprender computação paralela nos dias de hoje. A apresentação foi denominada como: "Computação Paralela: Despertando e Desplugando". Logo após esta aula inicial, realizou-se a interação dos estudantes com a maquete, que representa o algoritmo paralelo da soma de um vetor numérico. Durante a interação explicou-se o funcionamento do algoritmo paralelo da soma e os conceitos que o acompanham: A entrada do problema pode ser composta por dois vetores numéricos de tamanho oito ou um vetor de tamanho dezesseis. Na maquete proposta cada garrafa PET representa um processo durante a execução do algoritmo. Inicialmente, oito alunos de cada turma foram convidados a se posicionarem frente a uma garrafa que estivesse no nível mais alto da maquete (nível um). Para cada aluno foi entregue um *post it*, que o identificasse como um processo rotulado de zero a sete. Um instrutor, representando um processo mestre distribuiu então igualmente bolinhas de cor laranja para cada um dos alunos, que estavam em frente às garrafas. Estas bolinhas iniciais representam valores iguais dos vetores numéricos de entrada. Na proposta da maquete, cada cor de bolinha representa um valor determinado. Em seguida, os alunos foram convidados a soltarem as bolinhas, ao mesmo tempo, dentro das garrafas PET, que representavam os seus processos. Na maquete, cada garrafa contém três divisórias. A Figura 3 apresenta alguns dos momentos desta etapa inicial. Como próxima tarefa os alunos retiraram a primeira divisória da garrafa. Logo após, ao mesmo tempo, eles retiraram também a terceira divisória das garrafas azuis (nível um). Das garrafas azuis fluíram então bolinhas azuis, já previamente armazenadas, representando a soma das bolinhas de cor laranja. A partir destas ações, conceitos importantes de programação paralela foram discutidos com os estudantes, tais como: o número de processos escolhidos, a identificação de processos, a divisão de carga de trabalho, a computação local (operação de soma simples) e a sincronização de tarefas. Após a etapa inicial, os alunos observaram o fluxo de bolinhas azuis deslizando pelas canaletas para as garrafas do nível inferior. A partir dai alguns outros conceitos foram discutidos: a comunicação entre processos, a diminuição do número de processos ativos e a explicação do que é um passo de computação em um algoritmo paralelo. Além disso, por meio da maquete, os alunos perceberam que o número de garrafas-processos no nível dois é menor do que do nível um (metade), com isso, alguns processos não serão mais ativos. Os alunos que representavam processos inativos foram então convidados a deixar sua posição. As ações de manipulação das divisórias foram então repetidas pelos alunos-processos restantes até que apenas um aluno-processo, que representava a última garrafa ficou sozinho. Neste instante o último aluno foi capaz de calcular o valor da soma final sozinho e o disse ao instrutor. A Figura 4 apresenta alguns dos momentos desta fase final.

Figura 3. Etapas iniciais com as turmas de ensino técnico e superior, respectivamente.

Figura 4. Etapas intermediária e final com as turmas de ensino técnico integrado e superior em informática, respectivamente.

A estratégia da utilização da maquete proposta foi capaz de contribuir para desmistificar e ensinar alguns dos primeiros e difíceis conceitos introdutórios dos algoritmos paralelos. Durante as oficinas os alunos pareceram motivados e interessados com a metodologia proposta. Ao final das oficinas foi aplicado um teste para mensurar o conhecimento aprendido pelos estudantes. O teste foi composto de uma avaliação objetiva com dez questões de múltipla escolha. As questões foram relacionadas com assuntos abordados na oficina. O teste apresentou resultados de acertos bastante satisfatórios. A Figura 5 apresenta os gráficos com resultados do teste para as duas turmas. A primeira turma foi do primeiro ano do ensino médio integrado com informática composta por trinta e três alunos. A segunda turma do sexto semestre do curso de graduação em computação composta de dezessete alunos. Em ambas as turmas a quantidade de acertos no teste aplicado foi superior a setenta e oito por cento.

Figura 5. Resultado dos testes com as turmas de ensino técnico e superior



5. Conclusões

É um desafio atual e emergente motivar o ensino de programação paralela para os novos programadores. Diversas das metodologias convencionais aplicadas para este fim nas instituições de ensino, por vezes não são adequadas, seja pela linguagem adotada ou pela formato pedagógico aplicado. Desta forma é preciso que novas proposições de ensino sejam elaboradas e estimuladas, com intuito de diminuir a distância entre programadores paralelos iniciantes e seus tutores, por vezes, muito especializados.

Este trabalho apresentou uma proposta para o ensino de computação paralela, principalmente para programadores iniciantes, baseada em técnicas de computação desplugada. O público-alvo de nossa experiência foi formado por alunos de um curso técnico em informática e de um curso superior em computação. A partir da interação com a nossa maquete diversos conceitos iniciais e importantes em computação paralela puderam ser ensinados aos alunos e isto sem a utilização de computadores com *hardware* paralelo, que em muitos casos são bastante caros. Espera-se que este trabalho possa motivar mais pesquisa sobre novas abordagens para a educação em computação paralela, dada a carência do tema no Brasil. Como projetos futuros destaca-se a continuidade da oficina, por meio de aulas práticas, utilizando ferramentas e linguagens menos complexas para o ensino de computacão paralela, uma possibilidade é a utilização da linguagem XMT [Torbert et al. 2010]. Além disso pretende-se construir novas maquetes, que representem os algoritmos paralelos introdutórios das ementas curriculares, como os algoritmos de ordenação, de *list ranking*, entre outros. Os autores agradecem todo suporte oferecido pela Universidade Federal de Mato Grosso do Sul para realização deste trabalho.

Referências

Aureliano, V. C. O. and Tedesco, P. C. d. A. R. (2012). Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no sbie e wie. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 23.

Bachiega, N. G., S L Souza, P., Bruschi, S., Do, S., and S De Souza, R. (2017). Mapeamento sistemático do ensino teórico e prático de programação paralela. *Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (WALGPROG - WCBIE 2017)*.

Bell, T., Witten, I. H., Fellows, M., Adams, R., and McKenzie, J. (2011). Ensinando ciência da computação sem o uso do computador. *Computer Science Unplugged ORG*.

Bell, T. C., Witten, I. H., and Fellows, M. (1998). *Computer Science Unplugged: Off-line activities and games for all ages*. Citeseer.

Bogaerts, S. A. (2017). One step at a time: Parallelism in an introductory programming course. *Journal of Parallel and Distributed Computing*, 105:4–17.

Cambraia, A. C. and Scaico, P. D. (2013). Os desafios da educação em computação no brasil: um relato de experiências com projetos pibid no sul e nordeste do país. *Revista Espaço Acadêmico*, 13(148):01–09.

Cesar, Eduardo; Cortés, A. E. A. M. . M. J. C. S. A. S. R. (2017). Introducing computational thinking, parallel programming and performance engineering in interdisciplinary studies. *Journal of Parallel and Distributed Computing*, pages 116–126.

Costa, T., Batista, A., Maia, M., Almeida, L., and Farias, A. (2012). Trabalhando fundamentos de computação no nível fundamental: experiência de licenciandos em computação da universidade federal da paraíba. In *XX Workshop de Educação em Computação-WEI. Curitiba, PR, Brasil*.

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gregg, C., Tychonievich, L., Cohoon, J., and Hazelwood, K. (2012). Ecosim: a language and experience teaching parallel programming in elementary school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 51–56. ACM.

JáJá, J. (1992). *An introduction to parallel algorithms*, volume 17. Addison-Wesley Reading.

Júnior, J., Rapkiewicz, C. E., Delgado, C., and Xexeo, J. A. M. (2005). Ensino de algoritmos e programação: uma experiência no nível médio. In *XIII Workshop de Educação em Computação (WEI'2005). São Leopoldo, RS, Brasil*.

Kirk, D. and Hwu, W.-M. (2010). *Programando para processadores paralelos: uma abordagem prática à programação de GPU*. Elsevier Brasil.

Lessa, V., Forigo, F., Teixeira, A., and Licks, G. P. (2015). Programação de computadores e robótica educativa na escola: tendências evidenciadas nas produções do workshop de informática na escola. In *Anais do Workshop de Informática na Escola*, volume 21, page 92.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Prasad, S., Gupta, A., Kant, K., Lumsdaine, A., Padua, D., Robert, Y., Rosenberg, A., Sussman, A., and Weems, C. (2013). Literacy for all in parallel and distributed computing: guidelines for an undergraduate core curriculum. *The New Zealand Journal of Applied Computing and Information Technology*.

Prasad, S. K., Chtchelkanova, A. Y., Das, S. K., Dehne, F., Gouda, M. G., Gupta, A., JáJá, J., Kant, K., Salle, A. L., LeBlanc, R., Lumsdaine, M., Padua, D. A., Parashar, M., Prasanna, V. K., Robert, Y., Rosenberg, A. L., Sahni, S., Shirazi, B., Sussman, A., Weems, C. C., and Wu, J. (2011). Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: core topics for undergraduates. In *SIGCSE*.

Saule, E. (2018). Experiences on teaching parallel and distributed computing for undergraduates. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*.

Torbert, S., Vishkin, U., Tzur, R., and Ellison, D. J. (2010). Is teaching parallel algorithmic thinking to high school students possible?: one teacher's experience. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 290–294. ACM.

Vieira, A., Passos, O., and Barreto, R. (2013). Um relato de experiência do uso da técnica computação desplugada. *Anais do XXI WEI*, pages 670–679.

Vishkin, U. (1993). Thinking in parallel : Some basic data-parallel algorithms and techniques.

Wyllie, J. C. (1979). The complexity of parallel computations. Technical report, Ithaca, NY, USA.