Parallel Programming in Computing Undergraduate Courses: A Systematic Mapping of the Literature

F. Soares, C. Nobre, and H. de Freitas, *Member, IEEE*,

***Abstract*—Due to the current scenario in which multi-core architectures are predominant in most personal computers and servers, the knowledge of parallel programming content becomes fundamental for computer students to develop software capable of obtaining the best performance of these architectures. Considering the importance of this context, this paper presents the results of a systematic mapping of the literature related to the teaching-learning process of parallel programming in the computing programmes in three important databases: ACM, IEEE and Science Direct. The results obtained showed that in order to solve the challenges and differences found in teaching-learning parallel programming, reorganization is necessary in the undergraduate programmes. A standard for parallel programming teaching is important. This can be established by defining where and how to insert parallelism in the courses, adopting a methodology to teach the contents of parallelism in several different courses, beginning the study in the first year. The main languages, libraries, difficulties encountered and methods of classroom and distance teaching for parallel programming are presented in this paper. Distance learning is still little explored in this area of knowledge, but it can support the teaching and study of these contents.**

***Index Terms*—Learning, Parallel Programming, Distributed Computing, Teaching, Undergraduate.**

I. Introdução

GRANDES empresas de *hardware* buscam incrementar melhorias em seus componentes eletrônicos a fim de melhorar o desempenho dos processadores, diminuindo o tempo gasto ao executar programas. Alternativas que melhoravam a velocidade individual do processador do computador foram desenvolvidas e exploradas até esbarrarem em limitações físicas [18].

Uma das soluções encontradas foi projetar processadores, que até então possuíam somente um núcleo, com dois ou mais núcleos. Assim, a arquitetura conhecida como *single-core* (somente um núcleo) foi sendo substituída pela arquitetura conhecida como *multi-core* (mais de um núcleo). Essa arquitetura é hoje usada na maioria dos computadores de uso doméstico, estações de trabalho, servidores, *clusters* e outros [38].

Portanto, a execução de um programa, que até então só podia ser realizada por um processador *single-core*, passou a ser compartilhada com dois ou mais núcleos de processamento (i.e. processador *multi-core*). Assim, o esforço que seria gasto por um núcleo para executar determinado programa foi reduzido, explorando o potencial existente em múltiplos núcleos.

Segundo Brown et al. [18], diante da importância dessas arquiteturas, torna-se indispensável para os alunos de graduação das áreas de computação (Ciência da Computação, Engenharia de Computação, Engenharia de *Software* e Sistemas de Informação) saberem programar, de forma satisfatória, tanto em sequencial como em paralelo.

O convívio dos alunos com paralelismo desde os períodos iniciais do curso é muito importante [56], pois os alunos irão vê-lo como uma parte natural e comum da programação (da mesma forma que acontece com a programação sequencial), em vez de um conteúdo avançado e raramente usado [16]. Além disso, trocar o pensamento de sequencial para paralelo é uma tarefa difícil [56].

Existe um consenso geral de que os temas de paralelismo devem ser distribuídos ao longo do currículo de graduação [10]. Porém, na maioria das graduações isso não é possível, assim os conceitos de paralelismo são estudados somente nos últimos anos do curso e apenas alguns conteúdos são incluídos em cursos obrigatórios [3].

Um meio que pode ser utilizado para facilitar o acesso dos alunos aos conteúdos é o uso de portais educacionais, permitindo ao aluno estudar os conteúdos sem comprometer a rotina diária, adequando o estudo ao seu tempo e possibilitando o acompanhamento do professor [53]. Pesquisas para o contexto do ensino de programação de computadores usando educação a distância são necessárias, pois poucos trabalhos focam nessa abordagem de ensino [79].

Para orientar novas pesquisas na área do ensino de programação paralela nas graduações é importante o estudo dos trabalhos relacionados existentes e das experiências reportadas que consideram as dificuldades e desafios encontrados nesse contexto, os países e os anos que foram realizadas as pesquisas, em qual momento do curso foi sugerido iniciar esse conteúdo, as metodologias e ferramentas usadas para esse ensino e as metodologias adotadas quando o ensino desse conteúdo é realizado a distância.

O mapeamento sistemático da literatura é um método para identificar todas as pesquisas relevantes para uma determinada questão de pesquisa. A adoção desse método é importante para fornecer uma ampla visão geral da área pesquisada, estabelecendo se existem evidências de pesquisa sobre um tópico e identificando a quantidade dessas evidências [55].

Diante do exposto, há a necessidade de organizar um mapeamento sistemático da literatura com o objetivo de analisar as produções científicas referentes ao ensino-aprendizagem (presencial e a distância) de programação paralela nas graduações

Felipe Soares é mestrando em Informática pelo Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: falsoares@sga.pucminas.br.

Cristiane Nobre é professora do Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: nobre@pucminas.br.

Henrique Freitas é professor do Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: cota@pucminas.br.

na área de computação. Para isso, foi realizada uma pesquisa em trabalhos publicados nas bases *Association for Computing Machinery* (ACM), *Institute of Electrical and Electronics Engineers* (IEEE) e *Science Direct*, além de utilizar a ferramenta de busca *Scholar Google*.

O restante deste texto está organizado da seguinte maneira: a Seção II apresenta os trabalhos correlatos, a Seção III apresenta uma visão geral sobre programação paralela e sua correlação com programação concorrente e distribuída, a Seção IV apresenta o método usado no mapeamento, além das questões de pesquisa e os artigos selecionados. A Seção V apresenta os resultados e discussões. Por fim, a Seção VI conclui o trabalho e apresenta sugestões de trabalhos futuros.

II. Trabalhos Correlatos

Além de apresentar o panorama atual referente ao ensino-aprendizagem de paralelismo nos cursos de graduação, a contribuição deste trabalho está no fato de não existir mapeamento sistemático da literatura ou revisão sistemática da literatura que aborde os assuntos tratados neste trabalho. Alguns trabalhos apresentam um mapeamento ou revisão referente ao ensino-aprendizagem de programação [11][17], porém sem discutir sobre programação paralela. Outros trabalhos realizaram mapeamento sobre o ensino de programação paralela [13] e revisão de supercomputadores na educação superior [34]. Porém, este artigo apresenta um mapeamento mais amplo e discute um panorama mais completo sobre o tema ensino-aprendizagem de programação paralela nas graduações.

Bachiega et al. [13] realizaram um mapeamento sistemático do ensino de programação paralela, selecionando 24 artigos e analisando somente quais foram as estratégias de ensino e as ferramentas utilizadas. Já Fernández et al. [34] realizaram uma revisão em 34 estudos com o objetivo de caracterizar o cenário relacionado ao treinamento e educação em supercomputação, identificando as limitações e quais fatores podem melhorar esse estudo.

Além de discutir as questões propostas em Bachiega et al. [13] e Fernández et al. [34] (estratégias de ensino, ferramentas utilizadas, limitações e fatores que podem melhorar a aprendizagem e o ensino de programação paralela), este trabalho selecionou 83 artigos visando apresentar um panorama sobre o ensino de programação paralela nos cursos de graduação em computação, definindo também em quais cursos e em quais períodos o estudo foi proposto, a localidade das universidades que realizaram a pesquisa e as principais dificuldades encontradas pelos alunos, professores e instituições. Além disso, este trabalho pesquisou como está o cenário da educação a distância no ensino de programação paralela. Assim, este trabalho se difere dos correlatos nos artigos selecionados e nas questões de pesquisa analisadas.

III. Programação Paralela

Alguns conceitos importantes sobre o tema Programação Paralela [62][73], apresentados neste artigo, referem-se também a programação concorrente e programação distribuída. Nesse sentido, esta seção apresenta uma rápida discussão de cada um desses conceitos em um contexto de arquiteturas paralelas.

Programação concorrente é uma técnica que tem como objetivo a redução de dependência entre trechos de um mesmo código, possibilitando assim a execução fora de ordem destes trechos por processos e/ou *threads*. Porém, essa execução não necessariamente é em paralelo, uma vez que o paralelismo real de execução depende da arquitetura paralela de um computador, de processadores com suporte a *multithreading* ou *thread level speculation*, e/ou pelo uso de bibliotecas de paralelismo (i.e., programação paralela).

A programação paralela explora o potencial concorrente que existe no código via uso de bibliotecas de paralelismo que possibilitam a execução explícita de processos e/ou *threads* para execução simultânea. Nesse caso, é fundamental que exista uma arquitetura paralela (e.g., um processador *multi-core*) para que de fato o suporte ao paralelismo de execução ocorra. Por vezes, os termos concorrente e paralelo são usados como sinônimos, mas é importante ressaltar, que concorrência não implica em simultaneidade, mas em potencial de simultaneidade. Portanto, programação concorrente implica em identificar e reduzir dependências entre trechos de um mesmo código e a programação paralela implica em utilização de bibliotecas de paralelismo para exploração destes trechos em arquiteturas paralelas.

A programação distribuída é uma técnica utilizada para aumento de escalabilidade da execução do código paralelo fazendo uso de uma rede de comunicação. Nesse contexto, é necessário que os códigos paralelos façam uso de passagem de mensagem para comunicação coletiva, ou seja, a utilização de primitivas tais como *send* e *receive*. Não necessariamente a programação distribuída é utilizada para paralelismo, uma vez que o processamento pode ser distribuído e sequencial, mas no contexto deste artigo, o potencial distribuído de um sistema de computadores em *cluster*, oferece condições de exploração do paralelismo em uma escala maior. Por este motivo, o conceito programação paralela e distribuída também é utilizado para sistemas em que os vários nós com arquiteturas paralelas (e.g., múltiplos núcleos) estão interconectados por uma rede de comunicação de alto desempenho. Neste tipo de arquitetura em *cluster*, a programação paralela é realizada por memória (variável) compartilhada (comunicação intra-nó) e por passagem de mensagem em rede (comunicação inter-nós).

IV. Método

O método de mapeamento sistemático da literatura é um meio de identificar, avaliar e interpretar todas as pesquisas relevantes para uma determinada questão de pesquisa, área de tópico ou fenômeno de interesse. Para isso, são realizados levantamentos de dados, seguidos pela seleção dos artigos realizada por critérios de inclusão ou exclusão. Essas questões levantadas e os critérios de seleção definidos no início da pesquisa são chamados de protocolo de pesquisa [55].

A. Questões de Pesquisa

Este trabalho tem questão central de pesquisa: qual o panorama atual das publicações científicas sobre o processo

de ensino-aprendizagem de programação paralela nos cursos de graduação em computação? Para responder essa pergunta, foram definidas as seguintes questões de pesquisa.

QP1 Em qual momento do curso (ano) foi proposto o estudo de programação paralela nos cursos de graduação na área de computação?

QP2 Quais são os desafios encontrados pelos alunos e professores no ensino e aprendizagem de programação paralela nos cursos de graduação na área de computação?

QP3 Quais metodologias, linguagens, bibliotecas, interfaces de programação de aplicações e ambientes de desenvolvimento de programação paralela são usados na formação dos alunos no estudo de paralelismo?

QP4 Quais metodologias de ensino a distância têm sido utilizadas no ensino e aprendizagem de programação paralela?

Essas perguntas foram definidas para identificar o cenário atual do ensino de paralelismo nas graduações na área de computação, definindo os desafios encontrados para ensino e aprendizagem, qual momento do curso foi proposto estudar paralelismo, quais são os recursos, programas e didáticas usados no ensino e como está o cenário quando o ensino desse conteúdo é realizado a distância. Por meio desse estudo verificaram-se as produções anuais relacionadas ao tema e como elas evoluíram durante os anos, as instituições e os países que estão produzindo estudos nessa área.

B. Seleção dos Artigos

Os artigos selecionados foram encontrados nas pesquisas realizadas nas bases da ACM, IEEE e *Science Direct*, em agosto de 2019, cujo título ou resumo foram encontrados a partir da seguinte *string* de busca:

S1 (ensino OU educação OU aprendizagem) E (programação paralela OU computação paralela OU computação distribuída OU paralelismo) E graduação E ((desafios OU dificuldades) OU (modelos OU metodologias OU bibliotecas OU linguagens) OU (educação a distância OU ensino a distância))

A *string* também foi pesquisada em inglês:

S2 (*teaching* OU *education* OU *learning*) E (*parallel programming* OU *parallel computing* OU *distributed computing* OU *parallelism*) E (*undergraduate* OU *student*) E ((*challenges* OU *difficulties*) OU (*models* OU *libraries* OU *languages* OU *methodology*) OU (*distance education* OU *distance learning*))

A pesquisa foi realizada sem considerar a data de publicação dos artigos como filtro, pois mesmo os artigos mais antigos podem apresentar resultados importantes. Assim, foi encontrado um total de 279 artigos.

A Tabela I mostra a quantidade de artigos encontrados a partir das *strings* de busca em cada base pesquisada.

A próxima etapa foi a eliminação de artigos duplicados, e, posteriormente, foi realizada a leitura dos resumos dos artigos para seleção, descartando os trabalhos que não colaboravam com a pesquisa. Um total de 95 artigos foi selecionado, e, por fim, todos foram lidos e posteriormente os critérios de exclusão foram pesquisas antigas com informações desatualizadas ou artigos derivados da mesma pesquisa publicados em anos diferentes.

TABELA I
QUANTIDADE DE ARTIGOS ENCONTRADOS POR BASE DE DADOS

Base	Quantidade de artigos
ACM	109
IEEE	54
Science Direct	116
Total	279

Após todos os critérios adotados, restou um total de 83 artigos para a realização do mapeamento sistemático da literatura.

V. RESULTADOS E DISCUSSÃO

A Tabela II mostra a quantidade de trabalhos encontrados em cada curso para propor o ensino de programação paralela. O curso que mais tem discutido o assunto é Ciência da Computação, possuindo mais da metade das pesquisas encontradas. Há também estudos (25,3%) que não definiram o curso para qual propuseram o estudo.

TABELA II
QUANTIDADE E PORCENTAGEM DE PESQUISAS POR CURSO DEFINIDAS PARA ENSINO DE PROGRAMAÇÃO PARALELA

Curso	Quantidade de pesquisas	Porcentagem
Ciência da Computação	47	56,62%
Engenharia de Computação	6	7,22%
Engenharia de *Software*	3	3,61%
Sistemas de Informação	0	0%
Outros cursos	6	7,22%
Não definido	21	25,3%

A Fig. 1 ilustra a relação da quantidade dos artigos com cada ano de publicação. Levando em consideração que o primeiro processador *multi-core* foi lançado no mercado convencional no ano de 2004 [45] e que os artigos relacionados na Fig. 1 abordam o tema ensino de programação paralela, o aumento da quantidade de estudos posteriores ao ano de 2004 é muito significativo. A maioria das pesquisas concentra-se após o ano de 2012, uma hipótese para isso é o fato do período 2004 a 2012 ser para estudo dos processadores *multi-core*, definição de quais conteúdos seriam ensinados, troca de equipamentos nas universidades de pesquisa e outros.

Os estudos anteriores a 2004 e alguns estudos posteriores a esse ano são relativos ao ensino de paralelismo realizado por troca de mensagens em *clusters*. Porém, a grande maioria das pesquisas concentra-se no ensino de programação paralela para processadores *multi-core* com memória compartilhada ou abordam ambos os tipos de programação.

Em relação aos países onde ficam localizadas as universidades que realizaram as pesquisas, a maioria foiEstados Unidos da América (42 pesquisas), representando mais que 50% do total dos trabalhos. Posteriormente, apareceram Espanha (9 pesquisas), Nova Zelândia (5 pesquisas), Alemanha (4 pesquisas), Brasil e Índia (3 pesquisas cada). O trabalho [9]

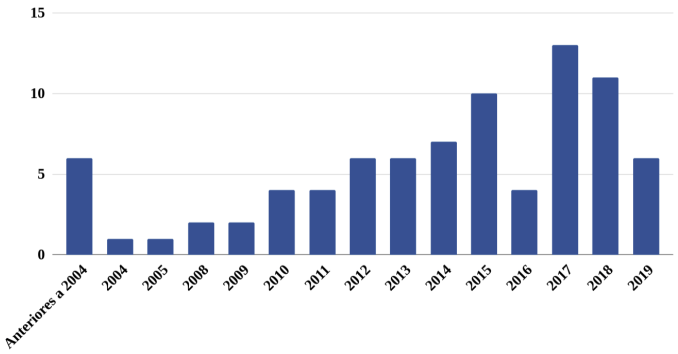

Fig. 1. Distribuição temporal dos artigos.

foi realizado tanto em uma universidade dos Estados Unidos da América, quanto em uma universidade da Austrália. A Tabela III ilustra a quantidade de pesquisas em cada país e quais são elas.

TABELA III
QUANTIDADE E PESQUISAS POR PAÍSES QUE FICAM LOCALIZADAS AS UNIVERSIDADES QUE AS REALIZARAM

País	Quantidade de pesquisas	Estudos primários
Alemanha	4	[39][51][54][84]
Argentina	1	[10]
Austrália	2	[9][56]
Brasil	3	[38][45][86]
Canadá	2	[1] [29]
China	1	[92]
Escócia	1	[40]
Espanha	9	[3][12][14][15][22][26][59][66][71]
Estados Unidos da América	42	[4][5][6][8][9][16][18][21][27][28][30] [31][32][33][35][36][37][47][48][49] [50][52][60][63][64][65][67][68][69] [70] [72][74][75][76][80][82][83][88] [89][90][93][94]
Índia	3	[23][57][61]
Nova Zelândia	5	[2][41][42][43][44]
Paquistão	2	[77][78]
Portugal	1	[25]
Russia	2	[7][81]
Sri Lanka	2	[85][91]
Suécia	1	[24]
Suíça	2	[20][46]
Taiwan	1	[58]

Somente 4 artigos foram realizados em universidades localizadas em países da América Latina, 3 artigos no Brasil [38][45][86] e 1 na Argentina [10], representando 4,82% do total dos artigos, enquanto o restante do mundo representa a grande maioria: 95,18%. Esse fato reforça que há poucos estudos publicados na área em países localizados na América Latina em comparação com o restante do mundo. No entanto, é importante ressaltar que esses dados não refletem uma ausência de pesquisas da comunidade latina americana, apenas refletem a não publicação dessas em veículos especializados. Por outro lado, é possível que publicações no prelo também não apareçam nas respostas, tal como Vasconcelos et al. [87] com publicação para 2019, que apresenta resultados do ensino de programação paralela para estudantes do primeiro ano de um curso de graduação em Ciência da Computação.

Nesta seção, são apresentadas também as respostas às questões de pesquisa deste mapeamento sistemático da literatura.

QP1 – Em qual momento do curso (ano) foi proposto o estudo de programação paralela nos cursos de graduação na área de computação?

Dentre os trabalhos que propuseram oferecer cursos para o ensino de programação paralela, um total de 37 artigos descrevem em qual período as turmas de ensino estavam matriculadas ao iniciarem o estudo. Os demais artigos não descrevem esse período. A Tabela IV mostra a quantidade de pesquisas para cada momento em que paralelismo foi ensinado, conforme conteúdo dos artigos.

TABELA IV
QUAL MOMENTO DO CURSO FOI PROPOSTO O ESTUDO DE PROGRAMAÇÃO PARALELA

Momento do curso	Quantidade de pesquisas	Porcentagem em relação ao total de pesquisas que definiram em qual momento iniciar o estudo
Períodos iniciais	15	40,54%
Ano 1	6	16,21%
Ano 2	9	24,32%
Ano 3	2	5,41%
Ano 4	1	2,7%
Ano 5	4	27,03%

Vale ressaltar que os trabalhos analisados neste artigo abordam pesquisas diferentes da realidade vivida na maioria das universidades, onde o ensino de paralelismo ocorre somente nos períodos finais dos cursos [3]. A proposta desses trabalhos é justamente quebrar esse paradigma e analisar a eficiência de ensinar paralelismo nos períodos iniciais e ao longo de todo o curso.

A maioria (40,54%) dos artigos realizou o estudo de programação paralela nos períodos iniciais dos cursos de graduação, porém não definiu em qual ano começar o estudo do conteúdo [18][31][32][33][37][38][67][69][70][72][76][83][88][93][94]. Alguns trabalhos [16][40][46][56][74][86] realizaram o estudo no primeiro ano do curso (1º e 2º semestres). Outros trabalhos [3][22][41][47][48][57][61][71][75] ofereceram o estudo no segundo ano (3º e 4º semestres). Chaudhury et al. [23] e Eccles, Nonneck e Stacey [29] realizaram o estudo no terceiro ano (5º e 6º semestres). Muresano, Rexachs e Luque [66] fizeram o estudo no quarto ano (7º ou 8º semestres). Por fim, Cunha e Lourenço [25], Giacaman e Sinnen [42] [44] e Ngo, Duffy e Apon [68] realizaram o estudo no quinto ano (9º ou 10º semestres).

Considerando os períodos iniciais como 1º e 2º ano, a maioria predominante (81,07%) das pesquisas realizou os estudos no início do curso, quebrando o paradigma que a maioria das universidades segue ao ensinar somente nos últimos períodos. Isso mostra que os pesquisadores estão conscientes

da importância do tema e que o mesmo deve ser explorado o quanto antes possível nos períodos iniciais [18][86].

Os trabalhos que defendem o ensino nos períodos iniciais sugerem que o aluno deve estudar o conteúdo ao longo de todo o curso começando por esses períodos, para assim tratar o ensino de paralelismo como um conteúdo natural e comum da programação, ao invés de um conteúdo difícil e raramente usado [16][18][56][86].

Já as pesquisas que realizaram o estudo nos períodos finais possuem características que justificam isso: Cunha e Lourenço [25] realizaram a pesquisa no ano de 1998 e por isso ainda seguiam uma grade curricular, metodologias e tecnologias antigas; Giacaman e Sinnen [44] e Giacaman e Sinnen [42] propuseram o estudo na disciplina de Engenharia de *Software* em um curso de Engenharia Elétrica, sendo que essa disciplina acontece nos períodos finais do curso. Por fim, Ngo, Duffy e Apon [68] sugeriram que o curso proposto deveria ser ofertado aos períodos iniciais, porém, nessa situação estudada no artigo, a maioria dos alunos realizam a disciplina nos períodos finais do curso.

QP2 – Quais são os desafios encontrados pelos alunos e professores no ensino e aprendizagem de programação paralela nos períodos iniciais dos cursos de graduação na área de computação?

De acordo com Udugama, Geeganage e Kuruppuarachchi [85], a programação paralela é um tópico complexo na área de computação. Assim, os professores e os alunos encontram vários desafios e dificuldades no ensino e aprendizagem dos conteúdos relacionados a esse tópico, os quais são descritos com mais frequência como:

- Alto custo dos equipamentos dos laboratórios para permitir o estudo prático de paralelismo [18][27][85][91];
- Pequeno número de professores com formação ou experiência em programação paralela, ou professores despreparados para ensino desse conteúdo [3][18][21];
- Necessidade de conhecimentos prévios, como por exemplo, arquitetura de computadores e sistemas operacionais [56][86];
- É necessário dedicar tempo para o estudo do algoritmo para aproveitar o potencial paralelo do código [86];
- Sistemas de *software*, interfaces de programação e ferramentas desconhecidas ou complexas [37];
- Os cursos da área de computação já possuem currículos cheios para inserir novos conteúdos [16][18];
- A transição do pensamento sequencial para o paralelo pode ser difícil [56][70];
- Programação paralela possui conteúdos e técnicas de programação avançados ou desafiantes [14][16][65][86].

Além dos principais desafios apresentados, foram citados com menos frequência:

- Pesquisadores com opiniões diversas sobre o modelo apropriado para computação paralela criaram abordagens diferentes para arquiteturas paralelas, linguagens e algoritmos associados [52];
- A ausência de um único livro didático que aborda programação paralela e seja referência na área [76];
- Ao projetar um curso, não é fácil encontrar um equilíbrio entre teoria e prática [36];
- Turmas heterogêneas (mistas em níveis de conhecimento), em que alguns alunos podem possuir conhecimento ou interesses distintos dentro de uma mesma sala de aula [15][94];
- A introdução de novos tópicos, particularmente em turmas novatas, implica em redução e/ou condensação de conteúdos [22];
- O processo de compilação e execução de um *software* paralelo é mais complicado do que a maioria dos compiladores para algoritmos sequenciais [37];
- O conjunto de habilidades exigidas depende do *hardware* paralelo particular que está trabalhando [43];
- Embora seja possível obter informações sobre conceitos de computação paralela pelo uso de simuladores, existem outros conceitos e questões que só podem ser apreciadas por meio de práticas com *hardware* verdadeiramente paralelo [76].

QP3 – Quais metodologias, linguagens, bibliotecas, padrões, modelos, plataformas, interfaces de programação de aplicações e ambientes de desenvolvimento de programação paralela são usados na formação dos alunos no estudo de paralelismo?

As principais metodologias propostas para o ensino de paralelismo foram:

- Acacio et al. [3], Ayguadé e González [12], Cunha e Lourenço [25], Mullen et al. [65] e Watkinson et al. [88] ofereceram o estudo de teorias para os alunos, seguido posteriormente de práticas que expõem os conceitos estudados na teoria e aproxima os alunos do *hardware* estudado. Por fim, os alunos realizaram testes para avaliarem o que foi aprendido e corrigirem os erros. Mullen et al. [65] denominaram esse modelo de "teoria e prática" e o citou como a melhor experiência educacional na área de ciência e engenharia. Esse modelo possui estudos desde 1998 até 2017, mostrando que o mesmo é usado desde pesquisas mais antigas até as atuais;
- Hyde [52] e Vasconcelos et al. [86] usaram simuladores com o intuito de ilustrar a relação entre programação paralela e arquitetura paralela por meio de um ambiente simples e viável. A data dos estudos varia de 1998 até 2017, mostrando que essa metodologia é explorada em pesquisas antigas e mais atuais;
- Adams, Brown e Shoop [4], Brown et al. [18] e Vasconcelos et al. [86] propõem estudar paralelismo durante todo o curso de programação, integrando os conteúdos nas disciplinas estudadas, como por exemplo: Algoritmo e Estrutura de Dados, Arquitetura de Computadores, Sistemas Operacionais, Compiladores, Algoritmos em Grafos e outras. Assim, os alunos têm contato com o conteúdo durante todo seu estudo, criando uma proximidade do estudante com o tema, da mesma forma que acontece com a programação sequencial. Brown et al. [18] e Vasconcelos et al. [86] denominaram esse modelo de "espiral", e as pesquisas que propuseram esse modelo

foram realizadas após o ano de 2010, sendo que todas sugeriram as linguagens de programação derivadas do C (C e C++) usando a API *OpenMP* para o ensino de paralelismo usando essa metodologia;

- Foley et al. [37] e Matthews [63] definiram um modelo que capacita os usuários por meio da exploração de algoritmos paralelos, expondo gradualmente mais detalhes sobre paralelismo e códigos paralelos. Assim, o aluno vai passando por uma série de módulos com conteúdos de níveis de conhecimento diferentes, ampliando e aprofundando esses conteúdos gradualmente. Esse modelo, chamado por Foley et al. [37] de "*bottom-up*", foi usado em pesquisas realizadas em anos posteriores a 2016;
- Foley et al. [37] sugeriram que os alunos programassem uma solução sequencial, e, posteriormente, sejam desafiados a trabalharem com um conjunto de dados muito grande para ser executado em sequencial. Assim, os alunos precisam pensar em como superar esse problema seguindo orientações relacionadas a paralelismo oferecidas pelo instrutor. Essa metodologia foi usada em uma pesquisa realizada no ano 2017 para o ensino de paralelismo;
- Georgi et al. [39] expõem uma série de bons conhecimentos de algoritmos em paralelo baseados em linguagens específicas e deixa o aluno livre para escolher qual tema deseja estudar e aprofundar. Esse estudo foi realizado em 2011.

Além de aprender os conteúdos de programação paralela, o aluno precisa estudar e conhecer o algoritmo que está paralelizando, pois um algoritmo mal paralelizado pode ter o tempo de execução superior a sua versão sequencial [45].

Os principais conteúdos sugeridos no ensino de paralelismo foram: analogias do paralelismo com a vida real, processos e *threads*, memória compartilhada e distribuída, concorrência, sincronização, padrões de comunicação, modelo *fork-join*, balanceamento de carga, condição de corrida, bloqueios, regiões críticas, exclusão mútua, sinais, *clusters*, análise de desempenho, eficiência e *speedup* [16][37][60][70].

O uso de práticas para o ensino de paralelismo é praticamente unânime nas metodologias adotadas nas pesquisas [3][48]. Quase todas as pesquisas usam ou se referem à prática como a melhor forma de oferecer o estudo desses conteúdos para os alunos, avaliando os desempenhos dos algoritmos e comparando-os com os sequenciais. Várias pesquisas também enfatizam que os alunos devem aprender a "pensar em paralelo" para resolverem os problemas computacionais [18][65]. Noções de arquitetura de computadores também são citadas como importantes para o aluno desenvolver as habilidades de paralelismo [45][56].

A Tabela V ilustra as linguagens de programação sugeridas para serem usadas no ensino de programação paralela. As linguagens *Fortran* (100%), *Occam* (40%) e C (8,33%) foram sugeridas em pesquisas mais antigas, anteriores ao ano de 1995. As outras pesquisas são todas posteriores a esse ano, sendo que, após 2015, C++ (25%) e *Java* (33,33%) reforçam uma tendência em pesquisas futuras. Por fim, todas os trabalhos que sugeriram *Python* ou *Snap* são posteriores a 2015, mostrando que essas também podem ter mais chances de serem usadas futuramente.

TABELA V
LINGUAGENS SUGERIDAS PARA O ENSINO DE PROGRAMAÇÃO PARALELA

Linguagens	Quantidade de pesquisas	Estudos primários
C	20	[4][5][8][15][27][33][44][45][54][56][58] [61][63][67][76][80][86][89][93][94]
C++	16	[10][18][22][24][45][46][47][51][54][58] [60][75][80][90][91][92]
Java	13	[5][9][16][22][41][42][46][47][50][54][77] [84][91]
Fortran	5	[15][52][65][76][91]
C#	3	[46][47][71]
Occam	3	[25][52][76]
Scratch	3	[18][31][33]
Python	3	[27][48][72]
Snap	2	[32][33]

Ao fazer a análise das linguagens sugeridas, as linguagens C e C++ encontram-se com maior número de sugestões. Observa-se ainda que mais da metade (57,35%) das pesquisas sugeriram a linguagem C ou derivadas do C (C++ e C#). A linguagem *Java* apareceu em 19,12%, *Fortran* em 7,35%, *Occam* em 4,41%, *Python* em 4,41%, e as linguagens de programação educacional por blocos *Scratch* e *Snap* apareceram em 4,41% e 2,94% respectivamente.

Além das linguagens de programação sugeridas para serem usadas no ensino de paralelismo, foram indicados Interface de Programação de Aplicações (API, do inglês *Application Programming Interface*) para Unidade Central de Processamento (CPU, do inglês *Central Process Unit*) e Unidade de Processamento Gráfico (GPU, do inglês *Graphics Processing Unit*), Ambientes de Desenvolvimento Integrado (IDE, do inglês *Integrated Development Environment*) modelos de programação, bibliotecas, padrões de comunicação e plataforma de *softwares*. A Tabela VI ilustra essas sugestões.

A API *OpenMP* foi a mais citada para ensino de programação paralela, pois suporta as linguagens C, C++ e *Fortran*, é simples de introduzir aos estudantes e grande parte do trabalho de paralelismo é feito de forma automática e implícita [18][45]. Além disso, todos os trabalhos realizados em universidades localizadas em países da América Latina [10][38][45][86] sugeriram utilizar as linguagens C ou C++ em conjunto com a API OpenMP, mostrando que nessa análise esses trabalhos possuem cenários similares a maioria dos trabalhos localizados no restante do mundo.

A API CUDA foi a mais sugerida para ensino de programação para GPU. O ensino de programação paralela em GPU com CUDA parece muito promissor, pois a maioria dos laboratórios das universidades já é adequada para o ensino prático de programação paralela e geralmente os alunos já estudaram C ou C++, linguagens de programação usadas para programar em CUDA [6].

O padrão de comunicação de dados em *clusters* mais citado foi MPI, tendo como principal motivação o fato de MPI ser escalável para milhões de núcleos e resolver muitos ou todos os problemas em um contexto *multi-core* [30].

TABELA VI
API, IDE, PADRÕES, MODELOS, PLATAFORMAS E BIBLIOTECAS USADOS NO ENSINO DE PARALELISMO

Nome	Descrição resumida	Quantidade de pesquisas	Estudos Primário
OpenMP	API para programação *multi-core*	28	[3][4][5][10][15][18][20][24][30][32][38][40][44][45][51][54][58][59][61][63][65][67][75][78][91][92][93][94]
MPI	Padrão para comunicação de dados em *clusters*	15	[3][9][15][20][38][51][54][58][63][65][69][72][77][78][94]
CUDA	API para GPU destinada a computação paralela GPU e computação heterogênea	8	[6][15][51][56][60][67][75][83]
OpenCL	API para ambientes computacionais heterogêneos (CPU, GPU e outros)	3	[10][54][56]
MapReduce	Modelo de programação e *framework* para suportar paralelismo em *clusters*	5	[27][54][58][64][69]
POSIX *threads*	Padrão POSIX para *threads*, o qual define uma API padrão para criar e manipular *threads*	2	[33][63]
Hadoop	Plataforma de *software* de computação distribuída voltada para *clusters*	3	[27][54][78]
Thread Building Block (TBB)	Biblioteca de modelos C++ para programação paralela em processadores *multi-core*	2	[47][58]
Simulador GEM5	Plataforma modular para pesquisa em arquitetura de sistemas completos	1	[78]
Visual Studio	IDE da *Microsoft* para desenvolvimento de *softwares*	1	[61]

QP4 - Quais metodologias de ensino a distância têm sido utilizadas no ensino e aprendizagem de programação paralela?

A educação a distância pode ser usada para auxiliar no ensino de programação paralela [23][37][48][51][81][65]. Essas quatro pesquisas representam 7,23% dos estudos referentes ao ensino de paralelismo, o que mostra que o ensino a distância e suas tecnologias são ainda pouco explorados nessa área do conhecimento. Além disso, todas as seis pesquisas foram publicadas após 2017, o que reforça que a adoção da educação a distância para o ensino dos conteúdos dessa área é recente.

Chaudhury et al. [23] desenvolveram a plataforma *web Let's HPC*[1] de acesso aberto em evolução para complementar a educação convencional dos conteúdos de computação de alto desempenho, permitindo que os usuários aprendam, avaliem, ensinem e vejam o desempenho de algoritmos paralelos. A metodologia adotada propõe usar a plataforma em 30 sessões de aula, 10 sessões de aulas práticas em laboratório e, por fim, os alunos são separados em duplas para criarem um projeto focado em escolher e compreender um algoritmo, escrever implementações sequenciais e paralelas desse algoritmo, seguidas de análise de desempenho teóricos e práticos.

Segundo Foley et al. [37], o projeto *OnRamp* propõe fornecer assistência no desenvolvimento das habilidades necessárias para programar em paralelo, por meio de uma série de níveis de conhecimento. Cada nível é projetado para ajudar os alunos por meio da aprendizagem de um conjunto de habilidades e conceitos necessários para usar um ambiente de computação paralela de forma natural. Esses quatro níveis representam uma possível progressão lógica do aprendizado que é necessário para escrever *software* paralelo: conceitos, paradigmas, linguagens e uso efetivo dos ambientes de computação paralela. O *OnRamp* usa uma abordagem *bottom-up*, onde o ambiente de aprendizagem baseado na *web* capacita os usuários expondo gradualmente mais detalhes sobre paralelismo e códigos paralelos.

Depois de pedir aos alunos uma solução sequencial, o professor introduz um conjunto de dados muito grande para ser executado em um computador pessoal (ou seja, precisa de mais memória ou leva horas ou dias para ser executado) e pede aos alunos que pensem sobre como eles superariam esse problema antes da próxima reunião da turma. O professor prepara uma implementação *multi-core* do mesmo programa e o organiza como um módulo *OnRamp*. O módulo é instalado, implantado em um ambiente de computação paralela e disponibilizado aos alunos por meio de um espaço de trabalho no portal *OnRamp*. O módulo permite que os alunos especifiquem os números de *threads*, tamanhos de dados e padrões de decomposição. A interface *web* contém observações feitas pelo professor e os parâmetros que os alunos podem especificar ao iniciar um trabalho.

Segundo Grossman et al. [48], o projeto COMP 322 foi originado em uma pesquisa desenvolvida a partir de coletas realizadas durante 6 anos, sobre o ensino de programação paralela nos segundos anos do curso de graduação em Ciência da Computação na Universidade de Rice, Estados Unidos. O COMP 322 é dividido em quatro etapas que ensinam os alunos a raciocinar sobre o comportamento de programas paralelos, educando-os tanto nas abstrações de alto nível, como também nos detalhes de trabalho com *threads* em *Java*.

Primeiramente, o COMP 322 oferece ferramentas teóricas para os alunos fundamentarem o comportamento e o desempenho de seus programas. Então, os alunos ganham experiência prática com estruturas de programação. Em seguida, os alunos têm a chance de usar estruturas de programação mais próximas do *hardware*, que são mais usadas na prática. Finalmente, os alunos recebem uma pesquisa de modelos alternativos e estruturas de outros campos em computação paralela.

Segundo Hundt, Schlarb e Schmidt [51], o *framework* para avaliação automática de códigos, chamado *System for Automated Code Evaluation* (SAUCE), foi desenvolvido com base em uma seleção de bons conhecimentos de algoritmos paralelos baseados em *threads* em C++, com uso de *OpenMP*, MPI e CUDA, que podem ser acrescentados em processadores de alto desempenho ou computação paralela.

Sukhoroslov [81] usou uma abordagem baseada no *software*

[1]www.letshpc.org. Acesso realizado em 04 de Setembro de 2019

Everest acessado usando um navegador *web* para introduzir tópicos e experiências práticas de paralelismo e computação distribuída. A plataforma foi utilizada pelos alunos para enviar códigos, colaborar na organização de exercícios e demonstrações mais práticas, e fornecer *feedbacks* imediatos aos alunos, reduzindo o tempo necessário para avaliação dos códigos.

Segundo Mullen et al. [65], as melhores experiências educativas nos domínios da ciência e da engenharia são construídas sobre o modelo teoria e prática. Assim, eles oferecem no ambiente *online* múltiplos vídeos (teoria), seguidos de conjuntos de problemas ou experiências (prática) e um teste ou projeto para sintetizar todos os conceitos e habilidades estudados. Avaliações frequentes podem reforçar a aprendizagem e descobrir equívocos antes de se fixarem para os alunos.

As seguintes ideias são propostas no curso: reorganizar os conteúdos relativos à programação paralela em módulos; criar exercícios e perguntas que levam os alunos a avançar no processo de aprender a pensar em paralelo; criar perguntas que reforcem as habilidades básicas necessárias para usar os sistemas de forma eficaz; fornecer caminhos que permitam ao aluno personalizar sua aprendizagem e criar uma comunidade *online* de praticantes para apoiarem uns aos outros em conjunto.

VI. Conclusão

Os resultados encontrados apresentam um panorama das publicações relacionadas ao tema ensino de programação paralela nos cursos de graduação na área de computação, encontradas nas bases de dados ACM, IEEE e *Science Direct*.

Quanto ao momento do curso em que foi proposto o estudo de programação paralela, a maioria predominante dos estudos, mais de 80%, considera importante iniciar os estudos de programação paralela nos períodos iniciais dos cursos de graduação, pois consideram a troca do pensamento sequencial para paralelo como difícil e acreditam que o estudo de paralelismo deve ser encarado em vários momentos do curso de graduação, para assim, tornar o conteúdo natural para os alunos.

Com relação aos desafios encontrados ao se propor o ensino e aprendizagem de programação paralela, pode-se observar que existem alguns desafios ao propor o ensino e aprendizagem desse conteúdo, porém a maioria pode ser solucionada com reorganização curricular nos cursos de graduação. Um padrão para o ensino de programação paralela é importante, e pode ser estabelecido definindo onde e como inserir paralelismo nas disciplinas dos cursos, adotando uma metodologia que ensina os conteúdos de paralelismo em várias disciplinas diferentes, começando o estudo desde os períodos iniciais.

Quanto as metodologias e as ferramentas de auxílio usados na formação dos alunos no estudo de paralelismo, as linguagens de programação para paralelismo mais citadas são C e C++, e as APIs mais citadas (*OpenMP* para programação *multi-core* e CUDA para computação paralela GPU) são integradas a essas linguagens. Várias metodologias diferentes são expostas nos artigos. As principais enfatizam a importância de ensinar programação em vários momentos do curso de graduação, expondo a teoria e posteriormente práticas em laboratórios. Várias pesquisas enfatizam a importância do aluno estudar e conhecer o algoritmo que está paralelizando, para assim conseguir obter o melhor desempenho na paralelização do mesmo.

Quanto a utilização da educação a distância, foram encontradas apenas seis pesquisas que usam educação a distância para ensino de programação paralela, o que mostra que o uso do ensino a distância e suas tecnologias é ainda pouco explorado nessa área do conhecimento. Além disso, todas as seis pesquisas foram publicadas após 2017, o que mostra que a adoção da educação a distância para o ensino dos conteúdos dessa área é recente. Todas as metodologias seguem o mesmo modelo das metodologias principais que são usadas na educação presencial. Porém, ao usar educação a distância, os alunos recebem conteúdos teóricos por meio de videoaulas e posteriormente são desafiados a realizarem práticas relacionadas ao conteúdo teórico estudado.

Além das questões propostas, foi apresentado um estudo sobre a relação dos cursos de graduação que oferecem o estudo de programação paralela, os anos de publicações dos artigos encontrados e a localidade das pesquisas. Com relação ao curso, observa-se que Ciência da Computação é o curso que mais recebe proposta de ensino desses conteúdos, seguido de Engenharia de Computação. Quanto aos anos das publicações, a maioria acontece após o ano de 2012, sendo 2017 o ano com a maioria das publicações. A maioria predominante das universidades que realizam as pesquisas desse tema está localizada nos Estados Unidos da América, seguido de Espanha, Nova Zelândia, Alemanha, Brasil e Índia.

Diate do exposto, a principal contribuição desse trabalho está em apresentar um mapeamento sistemático da literatura sobre o ensino e estudo de programação paralela nos cursos de graduação. Vários pontos importantes relacionados a esse contexto foram pesquisados e discussões foram apresentadas, podendo servir como base em trabalhos futuros que visam melhorar o panorama atual do contexto apresentado na pesquisa.

Portanto, como proposta de trabalho futuro, um estudo importante é criar um guia ou um currículo para o ensino de programação paralela. Esse guia poderia ajudar a definir onde e como inserir paralelismo nas disciplinas dos cursos de graduação na área de computação ou criar materiais, exercícios e uma metodologia para uma disciplina de programação paralela que será inserida no currículo do curso de graduação, principalmente nos períodos iniciais do curso. A educação a distância pode ser explorada para facilitar o ensino e estudo desses conteúdos.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores agradecem ao CNPq, à FAPEMIG, e à PUC Minas pelo suporte parcial na execução deste trabalho.

REFERÊNCIAS

[1] Abebe, M.; Glasbergen, B.; and Daudjee, K. "WatDFS: A Project for Understanding Distributed Systems in the Undergraduate Curriculum", in Proceedings of the 50th ACM Technical Symposium on Computer Science Education, ACM. Março, 2019.

[2] Abernethy, M. et al. "ParallelAR: An augmented reality app and instructional approach for learning parallel programming scheduling concepts.", in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, Vancouver, Canadá. Maio, 2018.

[3] Acacio, M. E. et al. "An experience of early initiation to parallelism in the Computing Engineering Degree at the University of Murcia, Spain", in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. Maio, 2012.

[4] Adams, J.; Brown, R. and Shoop, E. "Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to cs undergraduates", in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International. Maio, 2013.

[5] Adams, J. C.; Crain, P. A. and Stel, M. B. V. "TSGL a Thread Safe Graphics Library for Visualizing Parallelism", in Procedia Computer Science, 51, 1986 - 1995. ICCS. Dezembro, 2015.

[6] Anderson, N.; Mache, J. and Watson, W. "Learning CUDA: lab exercises and experiences", in Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, New York, NY, USA, pp. 183–188. Outubro, 2010.

[7] Antonov, A.; Popova, N.; and Voevodin, Vl. "Computational science and HPC education for graduate students: Paving the way to exascale", in Journal of Parallel and Distributed Computing, vol. 118, pp. 157-165. Agosto, 2018.

[8] Apon, A. et al. "Cluster computing in the classroom: Topics, guidelines, and experiences", in Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, pp. 476-483. Maio, 2001.

[9] Apon, A. et al. "Cluster computing in the classroom and integration with computing curricula 2001", in IEEE Transactions on Education 47.2, 188-195. Maio, 2004.

[10] Arroyo, M. "Teaching parallel and distributed computing to undergraduate computer science students", in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International, pp. 1297-1303. Maio, 2013.

[11] Aureliano, V. C. O. and Tedesco, P. C. d. A. R. "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), Vol. 23, n. 1. Novembro, 2012.

[12] Ayguadé, E.; and González, D. J. "An approach to task-based parallel programming for undergraduate students", in Journal of Parallel and Distributed Computing, vol. 118, pp. 140-156. Agosto, 2018.

[13] Bachiega, N. et al. "Mapeamento Sistemático do Ensino Teórico e Prático de Programação Paralela", in Anais dos Workshops do Congresso Brasileiro de Informática na Educação, Vol. 6, n. 1, pp. 1089. Outubro, 2017.

[14] Banchelli, F.; and Mantovani, F. "Filling the gap between education and industry: evidence-based methods for introducing undergraduate students to HPC.", in IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC), IEEE, Dallas, EUA. Novembro, 2018.

[15] Bernabé, G. et al. "A High Performance Computing Course Guided by the LU Factorization", in International Conference on Computational Science, 29, 1446 - 1457. Junho, 2014.

[16] Bogaerts, S. A. "One step at a time: Parallelism in an introductory programming course", in Journal of Parallel and Distributed Computing, vol. 105, pp. 4-17. Janeiro, 2017.

[17] Borges, R. P. et al. "A Systematic Review of Literature on Methodologies, Practices, and Tools for Programming Teaching", in IEEE Latin America Transactions, vol. 16, n. 5, pp. 1468-1475. Maio, 2018.

[18] Brown, R. et al. "Strategies for preparing computer science students for the multicore world", in Proceedings of the 2010 ITiCSE working group reports, ACM, New York, NY, USA, pp. 97-115. Junho, 2010.

[19] Budgen, D. and Brereton P. "Performing systematic literature reviews in software engineering", in Proceedings of the 28th international conference on Software engineering, New York, NY, USA, pp. 1051–1052. Maio, 2006.

[20] Burkhart, H.; Guerrera D. and Maffia, A. "Trusted high-performance computing in the classroom", in Proceedings of the Workshop on Education for High-Performance Computing, Piscataway, NJ, USA, pp. 27–33. Novembro, 2014.

[21] Burtscher, M. et al. "A module-based approach to adopting the 2013 ACM curricular recommendations on parallel computing", in Proceedings of the 46th ACM Technical Symposium on Computer Science Education, New York, NY, USA, pp. 36– 41. Fevereiro, 2015.

[22] Capel, M. I.; Tomeu, A. J. and Salguero, A. G. "Teaching concurrent and parallel programming by patterns: An interactive ICT approach", in Journal of Parallel and Distributed Computing, vol. 105, pp. 42-52. Julho, 2017.

[23] Chaudhury, B. et al. "Let's HPC: A web-based platform to aid parallel, distributed and high performance computing education", in Journal of Parallel and Distributed Computing, vol. 118, pp. 213-232. Agosto, 2018.

[24] Chozas A. C.; Memeti, S. and Pllana, S. "Using Cognitive Computing for Learning Parallel Programming: An IBM Watson Solution", in arXiv preprint arXiv:1704.01513, Zurich, Switzerland, vol. 108, pp. 2121-2130. Junho, 2017.

[25] Cunha, J. C. and Lourenço, J. A. "An integrated course on parallel and distributed processing", in ACM SIGCSE Bulletin, vol. 30, n. 1, pp. 217–221. Março, 1998.

[26] Daradoumis, T. et al. "Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming", in Computers & Education, vol. 128, pp. 158-170. Janeiro, 2019.

[27] Dingler, A.; and Bui, P. "Using workqueue to teach distributed computing", in Journal of Computing Sciences in Colleges, vol. 34, pp. 192-194. Outubro, 2018.

[28] Ebnenasir, A. and Mayo, J. "Fault-Tolerant Parallel and Distributed Computing for Software Engineering Undergraduates", in Parallel and Distributed Processing Symposium Workshop (IPDPSW), IEEE International, pp. 788-794. Maio, 2015.

[29] Eccles, R.; Nonneck, B. and Stacey, D. A. "Exploring parallel programming knowledge in the novice", in High Performance Computing Systems and Applications, HPCS, 19th International Symposium on, pp. 97-102 . Maio, 2005.

[30] Eijkhout, V. "Teaching MPI from mental models", in Proceedings of the Workshop on Education for High Performance Computing, IEEE Press, pp. 14–18. Novembro, 2016.

[31] Feldhausen, R.; Bell, S. and Andresen, D. "Minimum Time, Maximum Effect: Introducing Parallel Computing in CS0 and STEM Outreach Activities Using Scratch", in Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment, ACM, New York, NY, USA, pp. 75. Julho, 2014.

[32] Feng, A.; Gardner, M. and Feng, W. "Parallel programming with pictures is a Snap!", in Journal of Parallel and Distributed Computing, vol. 105, pp. 150-162. Julho, 2017.

[33] Feng, A.; Tilevich, E. and Feng, W. "Block-based programming abstractions for explicit parallel computing", in Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE, pp. 71-75. Outubro, 2015.

[34] Fernández, A. et al. "Supercomputers to improve the performance in higher education: A review of the literature", in Computers & Education vol. 128, pp. 353-364. Janeiro, 2019.

[35] Finlayson, I. et al. "Introducing tetra: an educational parallel programming system", in Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, pp. 746-751. Maio, 2015.

[36] Fisher, A. L. and Gross, T. "Teaching empirical performance analysis of parallel programs", in ACM SIGCSE Bulletin, ACM, vol. 24, n. 1, pp. 309-313. Março, 1992.

[37] Foley, S. S. et al. "OnRamp: A web-portal for teaching parallel and distributed computing", in Journal of Parallel and Distributed Computing, vol. 105, pp. 138-149. Julho, 2017.

[38] Freitas, H. C. "Introducing parallel programming to traditional undergraduate courses", in Frontiers in Education Conference (FIE), pp. 1-6. IEEE. Outubro, 2012.

[39] Georgi A. et al. "Linux cluster in theory and practice: A novel approach in teaching cluster computing based on the intel atom platform", in Procedia Computer Science, vol. 4, pp. 1917-1926. Maio, 2011.

[40] Ghafoor, S. K. et al. "Unplugged Activities to Introduce Parallel Computing in Introductory Programming Classes: an Experience Report", in Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ACM. Julho, 2019.

[41] Giacaman, N. "Teaching by example: using analogies and live coding demonstrations to teach parallel computing concepts to undergraduate students", in 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, pp. 1295-1298. Maio, 2012.

[42] Giacaman, N.; and Sinnen, O. "Preparing the software engineer for a modern multi-core world", in Journal of Parallel and Distributed Computing, vol. 118, pp. 247-263. Agosto, 2018.

[43] Giacaman, N.; Kalra, S. and Sinnen, O. "The active classroom: Students and instructors parallel programming in parallel", in Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, pp. 739-745. Maio, 2015.

[44] Giacaman, N. and Sinnen, O. "EA: Research-infused teaching of parallel programming concepts for undergraduate Software Engineering students", in Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, pp. 1099-1105. Maio, 2014.

[45] Gonçalves, R. et al. "Openmp is not as easy as it appears", in System-Sciences (HICSS), 2016 49th Hawaii International Conference on, pp. 5742-5751. Janeiro, 2016.

[46] Gross, T. R. "Breadth in depth: a 1st year introduction to parallel programming", in Proceedings of the 42nd ACM technical symposium on Computer science education, New York, NY, USA, pp. 435– 440. Março, 2011.

[47] Grossman, D. and Anderson, R. E. "Introducing parallelism and concurrency in the data structures course", in Proceedings of the 43rd ACM technical symposium on Computer Science Education, New York, NY, USA, pp. 505–510. Fevereiro, 2012.

[48] Grossman, M. et al. "Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level", in Journal of Parallel and Distributed Computing vol. 105, pp. 18-30. Julho, 2017.

[49] Harrell, S. L. et al. "Student cluster competition: a multi-disciplinary undergraduate HPC educational tool", in Proceedings of the Workshop on Education for High-Performance Computing, New York, NY, USA, pp. 4. Novembro, 2015.

[50] Humos, A. A. E. et al. "Incorporating PDC Modules into Computer Science Courses at Jackson State University", in Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, pp. 779-781. Maio, 2015.

[51] Hundt, C.; Schlarb, M. and Schmidt, B. "SAUCE: A web application for interactive teaching and learning of parallel programming", in Journal of Parallel and Distributed Computing, vol. 105, pp. 163-173. Julho, 2017

[52] Hyde, D. G. "A parallel processing course for undergraduates", in ACM SIGCSE Bulletin, ACM, vol. 21, n. 1, pp. 170–173. Fevereiro, 1989.

[53] Kaur, S. and Baba, M. S. "Development of a networking education portal for secondary education communities in Malaysia", in 2006 International Conference on Computing & Informatics, IEEE. pp. 1-6. Junho, 2006.

[54] Keller, R. "Teaching parallel programming to undergrads with hands-on experience", in Workshop on Parallel, Distributed, and High-Performance Computing in Undergraduate Curricula (EduPDHPC) in conjunction with Sc-13: The International Conference for High Performance Computing, Networking, Storage, and Analysis, pp. 1-8. Novembro, 2013.

[55] Kitchenham, B. and Charters, S. "Guidelines for performing systematic literature reviews in software engineering", in EBSE Technical Report EBSE-2007-01, Keele University and University of Durham, Vol. 2.3. Janeiro, 2007.

[56] Ko, Y.; Burgstaller, B. and Scholz, B. "Parallel from the beginning: The case for multicore programming in thecomputer science undergraduate curriculum", in Proceeding of the 44th ACM technical symposium on Computer science education, ACM, pp. 415-420. Março, 2013.

[57] Kumar, S. "oriented teaching of PDC topics in integration with other undergraduate courses at multiple levels: A multi-year report", in Journal of Parallel and Distributed Computing, vol. 105, pp. 92-104. Julho, 2017.

[58] Liu, P. et al. "Innovative system and application curriculum on multicore systems", in Proceedings of the 6th Workshop on Embedded Systems Education, ACM, New York, NY, USA, pp. 25–31. Outubro, 2011.

[59] López, P.; and Elvira, B. "Teaching high-performance service in a cluster computing course", in Journal of Parallel and Distributed Computing, vol. 117, pp. 138-147. Julho, 2018.

[60] Lupo, C.; Wood, Z. J. and Victorino, C. "Cross teaching parallelism and ray tracing: a project-based approach to teaching applied parallel computing", in Proceedings of the 43rd ACM technical symposium on Computer Science Education, ACM, pp. 523-528. Fevereiro, 2012.

[61] Manogaran, E. "ACt-PBL: An adaptive approach to teach Multi-core computing in University education", in Technology for Education (T4E), 2013 IEEE Fifth International Conference on, IEEE, pp. 19-23. Dezembro, 2013.

[62] Marlow, Simon. "Parallel and concurrent programming in Haskell". In Proceedings of the 4th Summer School conference on Central European Functional Programming School (CEFP'11), Viktória Zsók, Zoltán Horváth, and Rinus Plasmeijer (Eds.). Springer-Verlag, Berlin, Heidelberg, 339-401. 2011.

[63] Matthews, S. J. "Teaching with parallella: a first look in an undergraduate parallel computing course", in Journal of Computing Sciences in Colleges, vol. 31, n. 3, pp. 18-27. Janeiro, 2016.

[64] Matthews, S. J. "Using Phoenix++ MapReduce to introduce undergraduate students to parallel computing", in Journal of Computing Sciences in Colleges, vol. 32, n. 6, pp. 165-174. Junho, 2017.

[65] Mullen, J. et al. "Learning by doing, High Performance Computing education in the MOOC era", in Journal of Parallel and Distributed Computing, vol. 105, pp. 105-115. Julho, 2017.

[66] Muresano, R.; Rexachs, D. and Luque, E. "Learning parallel programming: a challenge for university students", in Procedia Computer Science, vol. 1, n. 1, pp. 875-883. Maio, 2010.

[67] Newhall, T.; Danner, A. and Webb, K. C. "Pervasive parallel and distributed computing in a liberal arts college curriculum", in Journal of Parallel and Distributed Computing, vol. 105, pp. 53-62. Julho, 2017.

[68] Ngo, L. B.; Duffy, E. B. and Apon, A. W. "Teaching HDFS/MapReduce systems concepts to undergraduates", in Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, IEEE, pp. 1114-1121. Maio, 2014.

[69] Ngo, L. B. et al. "Unifying computing resources and access interface to support parallel and distributed computing education", in Journal of Parallel and Distributed Computing, vol. 118, pp. 201-202. Agosto, 2018.

[70] Ontañón, S. et al. "Designing Visual Metaphors for an Educational Game for Parallel Programming", in Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems, ACM, New York, NY, USA, pp. 2818-2824. Maio, 2017.

[71] Ortin, F.; Redondo, J. M. and Quiroga, J. "Design and evaluation of an alternative programming paradigms course", in Telematics and Informatics, vol. 34, n. 6, pp. 813-823. Setembro, 2017.

[72] Ortiz-Ubarri, J.; Arce-Nazario, R. and Orozco, E. "Modules to teach parallel and distributed computing using MPI for Python and Disco", in Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International, IEEE, pp. 958-962. Maio, 2016.

[73] Pacheco, P. "An introduction to parallel programming", 1st ed., Morgan Kaufmann, Elsevier, 2011.

[74] Rague, B. "Measuring CS1 perceptions of parallelism", in Frontiers in Education Conference (FIE), IEEE, pp. S3E-1. Outubro, 2011.

[75] Rivoire, S. "A breadth-first course in multicore and manycore programming", in Proceedings of the 41st ACM technical symposium on Computer science education, ACM, New York, NY, USA, pp. 214-218. Março, 2010.

[76] Schaller, N. C. and Kitchen, A. T. "Experiences in teaching parallel computing—five years later", in ACM SIGCSE Bulletin, vol. 27 n. 3, pp. 15-20. Setembro, 1995.

[77] Shafi, A. et al. "Teaching parallel programming using java", in Proceedings of the Workshop on Education for High-Performance Computing, IEEE Press, pp. 56-63. Novembro, 2014.

[78] Shamsi, J. A.; Durrani, N. M. and Kafi, N. "Novelties in teaching high performance computing", in 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), IEEE, pp. 772-778. Maio, 2015.

[79] Silva, T. R. et al. "Teaching-learning of programming: a systematic literature review", in Brazilian Journal of Computers in Education, vol. 23, n. 1, pp. 182-196. Março, 2015.

[80] Speyer, G. et al. "Paradigms for parallel computation", in DoD HPCMP Users Group Conference, DOD HPCMP UGC, IEEE, pp. 486-494. Julho, 2008.

[81] Sukhoroslov, O. "Building web-based services for practical exercises in parallel and distributed computing", in Journal of Parallel and Distributed Computing, vol. 118, pp. 177-188. Agosto, 2018.

[82] Tabak, D. "ILP in the Undergraduate Curriculum", in Proceedings of the 2002 workshop on Computer architecture education: Held in conjunction with the 29th International Symposium on Computer Architecture, ACM, New York, NY, USA, pp. 18. Maio, 2002.

[83] Tran, Q. N. "Teaching design & analysis of multi-core parallel algorithms using CUDA", in Journal of Computing Sciences in Colleges, vol. 25, n. 4, pp. 7-14. Abril, 2010.

[84] Träff, J. L. "What the parallel-processing community has (failed) to offer the multi/many-core generation", in Journal of Parallel and Distributed Computing, vol. 69, n. 9, pp. 807-812. Setembro, 2009.

[85] Udugama, L. S. K.; Geeganage, J. and Kuruppuarachchi, W. V. "A configurable multi-core processor for teaching parallel processing", in Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on, IEEE, pp. 326-331. Dezembro, 2013.

[86] Vasconcelos, L. B. A. et al. "Ambiente para Estudo de Computação Paralela Baseado no Simulador Completo GEM5 e em Algoritmos de Ordenação Escritos com OpenMP", in International Journal of Computer Architecture Education (IJCAE), vol. 3, n. 1, pp. 1-4. Dezembro, 2014.

[87] Vasconcelos, L. et al. ”Teaching Parallel Programming to Freshmen in an Undergraduate Computer Science Program”, in 2019 IEEE Frontiers in Education Conference (FIE 2019), Cincinnati, USA. Outubro, no prelo 2019.
[88] Watkinson, N. et al. ”Teaching Parallel Computing and Dependence Analysis with Python.” in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, Rio de Janeiro, Brasil. Maio, 2019.
[89] Weeden, A. et al. ”The Blue Waters Petascale Institute: Longitudinal Impact and Assessment-Driven Development of an Intensive, Hands-on Curriculum for Teaching Applications in HPC”, in Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), ACM. Julho, 2019.
[90] Wein, J. et al. ”Virtualized games for teaching about distributed systems”, in ACM SIGCSE Bulletin, Chattanooga, TN, USA, vol. 41, n. 1, pp. 246-250. Março, 2009.
[91] Wepathana, Y. M. R. D.; Anthonys, G. and Udugama, L. S. K. ”Compiler for a simplified programming language aiming on Multi Core Students’ Experimental Processor”, in Industrial and Information Systems (ICIIS), 2015 IEEE 10th International Conference on, IEEE, pp. 284-289. Dezembro, 2015.
[92] Yang, J. et al. ”Using cP 2 BL in Teaching Multi-Core Related Contents”, in Young Computer Scientists, ICYCS 2008, The 9th International Conference for, IEEE, pp. 2449-2453. Novembro, 2008.
[93] Younis, A. et al. ”Case Study: Using Project Based Learning to Develop Parallel Programing and Soft Skills.” in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, Rio de Janeiro, Brasil. Maior, 2019.
[94] Zahran, M.; and Berger, M. J. ”Parallel Computing At The Undergraduate Level: Lessons Learned and Insights”, in Proceedings of the Workshop on Computer Architecture Education, ACM. Junho, 2019.

Felipe Augusto Lara Soares é graduado em Engenharia de Computação pela Pontifícia Universidade Católica de Minas Gerais (2016) e Técnico em Mecatrônica pelo CEFET-MG (2011). Atualmente é aluno de Mestrado em Informática da Pontifícia Universidade Católica de Minas Gerais. Tem interesse de pesquisa em Computação de Alto Desempenho, Programação Paralela, Análise de Dados, Descoberta do Conhecimento e Big Data.

Cristiane Neri Nobre possui graduação em Ciência da Computação pela Universidade Federal de Ouro Preto (1996), mestrado em Engenharia Elétrica pela Universidade Federal de Minas Gerais (1998) e doutorado em Bioinformática pela Universidade Federal de Minas Gerais (2007). É professora Adjunto IV do Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais. Atua principalmente nas seguintes áreas: Bioinformática, Interação Humano Computador e Inteligência Artificial.

Henrique Cota de Freitas possui graduação em Ciência da Computação (2000) e mestrado em Engenharia Elétrica (2003) pela Pontifícia Universidade Católica de Minas Gerais e doutorado (2009) em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Atuou como pesquisador convidado (2015 -2016) no grupo de pesquisa CORSE do INRIA Grenoble, França. Atualmente é professor da Pontifícia Universidade Católica de Minas Gerais. Tem interesse de pesquisa em computação de alto desempenho, computação heterogênea, arquiteturas paralelas e programação paralela.