UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Avaliação do Uso de Desafios no Aprendizado de Programação Paralela

Guilherme Martins

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Guilherme Martins

Avaliação do Uso de Desafios no Aprendizado de Programação Paralela

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
Julho de 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Martins, Guilherme
M386a Avaliação do Uso de Desafios no Aprendizado de Programação Paralela / Guilherme Martins; orientador Paulo Sérgio Lopes de Souza. -- São Carlos, 2020.
152 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Ensino. 2. Desafio de Programação. 3. Programação Paralela. I. de Souza, Paulo Sérgio Lopes, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Guilherme Martins

Evaluating Challenges Applied to Parallel Programming Learning

Dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
July 2020

Este trabalho é dedicado a todos que contribuíram, de alguma forma e em algum momento, para sua realização.

Em especial, dedico este texto a meus pais e irmãos, amigos, colegas do LaSDPC e ao meu orientador, Paulo Sérgio: um pesquisador de extrema competência, profissional de excelência e um ser humano admirável.

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais, por proverem minha educação e incentivarem toda minha trajetória acadêmica, que inclui a realização do presente trabalho. Agradeço aos meus irmãos, demais familiares e amigos por apoiarem minhas decisões e me motivarem a seguir este caminho. Agradeço aos colegas do LaSDPC pelo compartilhamento de conhecimento, recursos e pela amizade. Agradeço ao meu orientador, Dr. Paulo Sérgio Lopes de Souza, por permitir e apoiar a escrita deste trabalho, por meio de seus valiosos conselhos, profissionalismo exemplar e experiência extraordinária.

Agradeço também a Thiago de Jesus Oliveira Durães, pela contribuição na atualização do mapeamento sistemático descrito nesta dissertação e Gabriel Martins, pelo apoio na padronização da descrição e dos códigos fonte dos desafios utilizados neste trabalho, bem como pela implementação de alguns desafios extra que poderão ser utilizados em trabalhos futuros.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro à pesquisa. Agradeço também à Universidade de São Paulo, em especial ao Instituto de Ciências Matemáticas e de Computação, e seus servidores por proverem ferramentas, conhecimento e habilidades necessárias para a realização deste trabalho.

Em especial, agradeço à Ana Spengler e Davi Conte pelo fundamental suporte provido durante a realização deste trabalho.

“Life’s an education - a multiple equation,
everybody finds their own solution in the end.”
(Martin Walkyier)

RESUMO

MARTINS, G. **Avaliação do Uso de Desafios no Aprendizado de Programação Paralela**. 2020. 152 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

O aprendizado de programação paralela não é trivial devido à complexidade dos conceitos que a fundamentam, dificuldades de compreensão e visualização do comportamento dos programas concorrentes, e dependência de conhecimentos provenientes de outras disciplinas de ciência da computação. Tais fatores, unidos à ausência de um aprendizado mais agregador e à crescente demanda por profissionais capacitados em desenvolvimento de aplicações de alto desempenho, justificam a necessidade de novos métodos e recursos que facilitem o processo de ensino-aprendizado que favoreça o desenvolvimento das habilidades e competências esperadas neste contexto. O objetivo do presente trabalho é avaliar o uso de desafios para ensinar programação paralela, independentemente da metodologia de ensino (tradicional, *Problem Based Learning* e outras) ou sistemas de suporte a maratonas de programação. Analisamos como os desafios contribuem para o aprendizado da programação paralela, considerando aspectos técnicos e motivacionais. Os resultados mostram o aprendizado em termos de conteúdo teórico, qualidade e corretude de código. Além disso, representam o nível de satisfação dos estudantes em relação à qualidade do curso. Os resultados foram positivos em relação às análises feitas, evidenciados por percentagens de até 85% em qualidade de código e 83% em satisfação dos alunos. Concluímos que o uso de desafios de programação afeta positivamente o aprendizado de programação paralela, estimula o desenvolvimento de soluções criativas e promove um ambiente saudável de competição entre os alunos. Além disso, percebemos que o uso de sistemas de maratona de programação traz benefícios, como *feedback* imediato e avaliação simplificada, mas requer um esforço considerável dos responsáveis pelos cursos para preparar as aulas e manter a infraestrutura computacional.

Palavras-chave: Ensino, Desafio de Programação, Programação Paralela.

ABSTRACT

MARTINS, G. **Evaluating Challenges Applied to Parallel Programming Learning**. 2020. 152 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Learning parallel programming is not trivial, due to its basic concepts, difficulties to understand and view the behavior of concurrent programs, and the knowledge dependence from other computer science disciplines. Such factors, plus the absence of a fully effective learning model and the growing demand for trained professionals in the development of high performance applications, justify the need for new methods and resources capable to make easier the teaching-learning process, which favour the development of skills and competences expected for this context. The objective of this work is to evaluate the use of challenges to teach parallel programming, regardless of the teaching methodology (traditional, Problem Based Learning and others) or support systems for programming marathons. We analyze how the challenges contribute to the learning of parallel programming, considering technical and motivational aspects. The results show learning in terms of theoretical content, quality and correctness of code. In addition, they represent the level of student satisfaction with the quality of the course. The results were positive in relation to the analyzes made, evidenced by percentages of up to 85% in code quality and 83% in student satisfaction. We conclude that the use of programming challenges positively affects the learning of parallel programming, stimulates the development of creative solutions and promotes a healthy competition environment among students. In addition, we realized that the use of programming marathon systems has benefits, such as immediate textit feedback and simplified assessment, but it requires considerable effort to prepare classes and maintain the computational infrastructure.

Keywords: Teaching, Programming Challenge, Parallel Programming.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* - Interface de Programação de Aplicativos
CPU	*Central Processing Unit* - Unidade de Processamento Central
GPU	*Graphics Processing Unit* - Unidade de Processamento Gráfico
HPC	*High Performance Computing* - Computação de Alto Desempenho
OER	*Open Educational Resource* - Recurso Educacional Aberto
PBL	*Problem Based Learning* - Aprendizado Baseado em Problemas
PBL	*Project Based Learning* - Aprendizado Baseado em Projetos
REST	*Representational State Transfer* - Transferência Representacional de Estado
TBL	*Team Based Learning* - Aprendizado Baseado em Equipes

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contexto

Computação paralela pode ser definida como o tipo de computação na qual vários cálculos e operações são realizadas simultaneamente, partindo do princípio de que problemas complexos podem ser subdivididos em problemas menores. Os problemas fracionados são processados paralelamente em vários recursos de processamento e, no fim, tem-se a reunião dos resultados intermediários, obtendo-se o resultado final (ALMASI; GOTTLIEB, 1989).

Consideram-se concorrentes dois ou mais processos que começaram suas execuções e em um determinado instante do tempo ainda não as terminaram. Processos paralelos são um tipo especial de processos concorrentes, os quais estão de fato em execução ao mesmo tempo, em recursos de processamento distintos (ALMASI; GOTTLIEB, 1989). Pode-se considerar, portanto, que os programas paralelos são especificações das implementações concorrentes, compartilhando todas as características dos mesmos. Desta forma, assume-se no presente trabalho, a igualdade entre a programação concorrente e paralela e de seus produtos (programas).

A computação de alto desempenho, comumente identificada pela sua sigla em inglês HPC, para *High Performance Computing*, refere-se ao uso de dispositivos computacionais de forma a alcançar melhores desempenhos para aplicações específicas. A programação paralela é muito usada para a HPC, sendo altamente valorizada e substancialmente importante para aplicações que requerem alto poder computacional e para sistemas cujo tempo de resposta é um atributo crítico (HAGER; WELLEIN, 2010). Recentemente, a importância dos conceitos da programação paralela para o desenvolvimento de soluções para problemas complexos, de áreas como engenharia, matemática, pesquisa operacional e outras, teve um crescimento significativo, sobretudo devido à evolução dos recursos de *hardware* e modelos de programação (FIORE; BAKHOUYA; SMARI, 2018). Devido à crescente presença e utilização de dispositivos *multi core* e *many core*, além do surgimento da computação heterogênea com Unidades de

Processamento Central (CPUs) e Unidades de Processamento Gráfico (GPUs)), o desempenho do *software* paralelo obteve melhoria considerável (BOURGOIN; CHAILLOUX; LAMOTTE, 2014). Entretanto, as metodologias de ensino referentes às disciplinas de programação paralela não apresentam, ainda, um modelo sólido para utilização eficiente dos referidos dispositivos (MURESANO; REXACHS; LUQUE, 2012).

A crescente importância da HPC justifica o surgimento de uma disciplina de mesmo nome, virtualmente equivalente às disciplinas de programação/computação paralela, porém com o foco no desenvolvimento de algoritmos e aplicações que objetivam melhores desempenho, principalmente focados na redução do tempo de resposta (SHAMSI; DURRANI; KAFI, 2015). Devido à similaridade entre os objetos de ensino, considera-se neste texto, também, a equivalência entre as disciplinas de Programação Concorrente, Programação Paralela e Computação de Alto Desempenho, além dos demais conteúdos de nomenclatura diferente, porém com ementa similar. É importante destacar, no entanto, que, no presente trabalho, Computação de Alto Desempenho não é tratada como uma disciplina ou conteúdo, mas como o conjunto de recursos, fundamentados nas disciplinas citadas anteriormente, que permitem o desenvolvimento de soluções computacionais, cujos requisitos de desempenho sejam atendidos para a aplicação ou contexto em que o mesmo está inserido.

Os Recursos Educacionais Abertos (REAs), ou do Inglês OER (Open Educational Resources), são materiais de apoio ao ensino que podem ser acessados e modificados livremente, geralmente desenvolvidos por um grupo acadêmico. Tratam-se de uma forma eficaz e democrática de disseminar o conhecimento (D'ANTONI, 2009). Podem ser considerados Recursos Educacionais Abertos, no contexto de computação, planos de aula, apresentações em *slide*, demais materiais de apoio teórico; *software* educacionais, tais como simuladores, *solvers*, jogos eletrônicos e *interfaces* gráficas ou quaisquer elementos de acesso livre e irrestrito, capazes de ilustrar e ampliar a compreensão sobre um tópico ou conceito de uma disciplina específica (DICHEV; DICHEVA, 2012).

O ensino de plataformas de computação apoiado pelo desenvolvimento e utilização de REAs obteve resultados importantes, ilustrados pelo aumento do desempenho dos alunos na produção de soluções criativas para problemas em temáticas consideradas complexas tais como sistemas operacionais, redes de computadores e sistemas distribuídos (CACHO *et al.*, 2016).

Existem estudos, como Almeida *et al.* (2012) e Giménez (2016), que demonstram o potencial do uso de maratonas de programação como recurso educacional, a fim de incentivar os alunos a desenvolver soluções de qualidade e alto desempenho. No entanto, tais experimentos têm forte dependência do uso de sistemas de maratona de programação e metodologias de ensino baseadas na solução de problemas, como a aprendizagem baseada em problemas ou aprendizagem baseada em projetos, o que limita a aplicabilidade das metodologias propostas e exige um esforço adicional considerável dos professores responsáveis para a preparação dos cursos e manutenção da infraestrutura computacional necessária.

O presente trabalho está inserido no contexto de desafios de programação para suporte ao ensino de programação paralela.

1.2 Motivação

Há uma demanda crescente por profissionais capacitados em desenvolver aplicações de alto desempenho, fundamentadas nos conceitos de programação paralela. Entretanto, há ainda fatores que dificultam o ensino e o aprendizado, tal como a complexidade dos conceitos e dificuldade em visualizar o funcionamento dos elementos internos aos programas paralelos (BOURGOIN; CHAILLOUX; LAMOTTE, 2014).

Devido aos recentes e favoráveis resultados na literatura acerca do uso de desafios e maratonas de programação no contexto de disciplinas de programação paralela, justifica-se a necessidade de desenvolver novas estratégias neste contexto, no intuito de facilitar o processo de ensino-aprendizagem (SHAMSI; DURRANI; KAFI, 2015).

1.3 Trabalhos Prévios

Uma das linhas de pesquisa exploradas por membros do Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC), trata-se do desenvolvimento de objetos de ensino, recursos educacionais e experimentos caracterizados pela adaptação e aplicação de abstrações de ensino-aprendizado no contexto das disciplinas relacionadas à computação de alto desempenho.

Durante a disciplina de Sistemas Operacionais (SSC0640), ministrada no primeiro semestre do ano de 2015 para os alunos do curso de Engenharia da Computação do ICMC-USP, foi proposto o desenvolvimento de recursos educacionais capazes de ilustrar os conceitos da disciplina. Sob orientação do professor responsável pela disciplina, foram implementados 14 REAs, que representam diversos tópicos do conteúdo de Sistemas Operacionais tais como processos, memória e sistemas de arquivo. Tais recursos podem ser obtidos através do site do LaSDPC, no endereço <http://rea.lasdpc.icmc.usp.br/en/oers/>.

Uma outra ferramenta desenvolvida pelo grupo de pesquisa constitui em um simulador dos conceitos de hierarquia de memória, vistos em disciplinas de Sistemas Operacionais, denominado Amnesia. O referido *software* designa um recurso educacional aberto capaz de ilustrar os conceitos de registradores, funcionamento da memória principal e estratégias de paginação. As características e arquitetura da ferramenta são melhor detalhadas no artigo de Cacho *et al.* (2016). O sistema encontra-se disponível no endereço <http://amnesia.lasdpc.icmc.usp.br/amnesia-en/>).

Foram desenvolvidos também REAs para demonstrar o funcionamento de *Firewalls* (<http://lasdpc.icmc.usp.br/~ssc640/firewall/inicial.html>) e do sistema de arquivos HDFS, da ferramenta Apache Hadoop (<http://lasdpc.icmc.usp.br/~ssc640/hadoop/>).

O trabalho de conclusão de curso realizado pela aluna de bacharelado em ciências de computação (BCC) Ana Caroline Spengler, orientanda do professor Dr. Paulo Sérgio Lopes de Souza, descreve a aplicação da metodologia de aprendizado colaborativo *Team Based Learning* (TBL), descrita na Seção 2.6.2, no contexto da disciplina de programação concorrente em uma turma de BCC. Tal estudo constitui em uma aplicação, até então inédita, da estratégia em questão em disciplinas de programação concorrente, que apresentou bons resultados, ilustrados pelo *feedback* positivo dos estudantes e desempenho na etapa de avaliação componente do TBL.

1.4 Problema

Aprender programação paralela não é trivial, devido à complexidade de seus conceitos base, dificuldades para entender e visualizar o comportamento de programas paralelos e à dependência de conhecimento derivado de outras disciplinas da ciência da computação. Tais fatores, somados à ausência de um modelo de aprendizado totalmente eficaz e a crescente demanda por profissionais treinados no desenvolvimento de aplicações de alto desempenho, justificam a necessidade de novos métodos e recursos capazes de facilitar o processo de ensino-aprendizagem, que favorecem o desenvolvimento de aptidões e competências esperadas para este contexto.

1.5 Hipótese

É possível aprender programação paralela por meio do uso de desafios, de forma em que as implementações paralelas que constituem na resolução destes desafios sejam corretas e possuam qualidade, em função do bom uso dos recursos de linguagem, modelo de programação e desempenho. Além disso, os alunos se sentem motivados a aprender por meio da abordagem proposta e desafiados a produzir soluções paralelas satisfatórias.

1.6 Objetivos

Objetivo geral deste trabalho é avaliar o uso de desafios de programação no aprendizado de programação paralela. Desta forma, visa-se mensurar o nível da absorção de conhecimento pelos alunos, a qualidade do código produzido por eles e a motivação dos estudantes quando submetidos ao uso de desafios de programação com diferentes metodologias de ensino.

1.7 Estrutura da dissertação

O presente trabalho está estruturado, em função da organização dos Capítulos, da forma descrita a seguir.

O Capítulo 2 descreve a fundamentação teórica para a realização do presente estudo, que constitui em uma síntese sobre abordagens, métodos e técnicas pedagógicas, além dos principais exemplares para cada categoria. Este capítulo detalha metodologias consideradas fundamentais para a realização deste trabalho, tais como o *Team Based Learning* e *Problem Based Learning*. Na porção final do Capítulo, apresentam-se conceitos sobre desafios e maratonas de programação, associados às principais tecnologias de apoio as suas realizações.

No Capítulo 3, faz-se um compilado sobre as diretrizes curriculares sobre o ensino de programação paralela definidas pela *ACM* e *IEEE*, além de principais técnicas e tendências aplicadas neste contexto, além das lacunas identificadas neste âmbito.

Por sua vez, o Capítulo 4 constitui em um mapeamento sistemático visando identificar o estado da arte em metodologias e recursos educacionais empregados no ensino e aprendizado de programação paralela, o que inclui a categorização das publicações encontradas por período e tipo de contribuição identificada, além da qualidade relativa dos estudos encontrados.

O Capítulo 5 descreve a realização de 4 experimentos, em função de uma metodologia para seu planejamento, condução e análise, descrita em Wohlin *et al.* (2012). Para cada um destes experimentos, explicitam-se seus resultados, em razão de aprendizado teórico, qualidade dos códigos desenvolvidos e grau de motivação dos alunos. No fim deste Capítulo, fazem-se as análises gerais buscando responder as questões de pesquisa definidas para cada experimento e os objetivos deste projeto de pesquisa.

Finalmente, o Capítulo 6 consiste na descrição e detalhamento do impacto dos resultados apresentados, contribuições obtidas, contribuições científicas, limitações desta dissertação e possíveis trabalhos futuros.

CAPÍTULO

2

REFERENCIAL TEÓRICO

2.1 Considerações Iniciais

No intuito de elencar os conceitos fundamentais para a compreensão e fundamentação teórica do presente trabalho traz-se, neste capítulo, em um primeiro momento, definições acerca da teoria pedagógica, no que se refere à terminologia praticada no contexto do ensino.

Em um segundo momento, descrevem-se metodologias de aprendizado ativo e colaborativo, suas características, técnicas e contexto de sua aplicação.

Finalmente, apresentam-se definições acerca de desafios e maratonas de programação, suas particularidades, utilização no contexto de ensino e principais sistemas de apoio à tal estratégia.

É válido destacar que os tópicos relacionados à teoria educacional aqui apresentados não são abordados em profundidade. O tratamento destes conceitos, no âmbito deste trabalho, é limitado ao contexto de ciência da computação, especificamente restrito às disciplinas-base da computação de alto desempenho.

2.2 Ensino e Aprendizado

O ensino pode ser definido como o processo de transmissão de conhecimento, valores, habilidades e atitudes, desejavelmente benéficos para um conjunto de indivíduos. Por esta definição, infere-se que o professor, como sujeito da transferência de conhecimento, é o elemento ativo do processo de ensino, sendo o estudante o objeto do mesmo (CAZDEN, 1988).

Pela definição inversa, o aprendizado trata-se do processo de adquirir conhecimento, desenvolver virtudes e habilidades. Neste processo, o aluno ou estudante é o sujeito da busca pelo conhecimento, sendo o professor o mecanismo facilitador para o aprendizado (CAZDEN,

1988).

No contexto deste trabalho, os conceitos de ensino e aprendizado são tratados em função dos papéis exercidos pelo professor e o aluno. Assume-se, portanto, que o ensino é centrado na participação do professor, como elemento ativo do processo; e o aprendizado é focado no papel do aluno, como sujeito ativo do procedimento da aquisição de conhecimento.

2.3 Metodologia de Ensino

Existem diversas definições sobre a terminologia referente ao processo de ensino. As expressões mais comumentes utilizadas e encontradas na literatura relacionada são método, metodologia e abordagem.

O presente trabalho será direcionado pela taxonomia proposta por ANTHONY (1963) que, embora trata-se de uma referência consideravelmente datada, traz uma definição satisfatória para a representação almejada por este texto. No referido artigo, o autor traz a definição hierárquica de abordagem, método e técnica como processos de ensino. O significado de cada um dos termos, segundo o autor, é descrito abaixo.

- Abordagem: Um conjunto de suposições acerca do que se deseja ensinar ou aprender (o que ensinar);
- Método: Um plano geral para apresentação sistemática do que se deseja ensinar, baseado na abordagem adotada (como ensinar);
- Técnica: Atividades específicas, consistentes com o método e abordagem, utilizadas para atingir um objetivo pontual. A técnica trata da especificação do método, por meio do detalhamento dos tópicos e dos meios utilizados no ensino.

A Figura 1 ilustra a hierarquia entre os termos proposta pelo autor.

No presente trabalho, considera-se o conceito de metodologia, similar à definição de método, como a forma de ensinar o conteúdo alvo, que trata-se, neste caso, de computação de alto desempenho. Assim, posteriores referências ao termo metodologia indicam o conjunto de técnicas que descrevem, coletivamente, o modo como pode ser ministrado o tópico em questão.

2.4 Metodologia Tradicional de Ensino

De acordo com Mizukami (1986), o ensino tradicional pode ser caracterizado pela concepção de educação como um produto, no qual ocorre a transmissão de ideias selecionadas e organizadas logicamente. No que se refere ao processo ensino-aprendizagem, destacam-se as

Figura 1 – Hierarquia da terminologia de ensino

Fonte: ANTHONY (1963).

situações em sala de aula, nas quais o aluno é meramente instruído e ensinado pelo professor, que por sua vez, torna-se o detentor do conhecimento.

Neste modelo, portanto, o aluno é passivo em relação ao seu aprendizado, de forma que este depende, exclusivamente, dos recursos oferecidos pelo professor, os quais devem ser adquiridos por meio da participação em aula, majoritariamente como mero ouvinte (SHOVEIN *et al.*, 2005).

A Figura 2 ilustra, de maneira cartunesca, o modelo de ensino tradicional, assim como o papel do professor e estudantes no contexto de sala de aula.

Ainda, segundo Mizukami (1986), o professor detém a autoridade sobre a definição do conteúdo e sobre o processo de comunicação, sendo seu papel restrito à transmissão do conhecimento predefinido. O aluno, por sua vez, tem a função de absorver os conteúdos oferecidos e de reproduzi-los, muitas vezes de forma automática.

Embora ainda seja o modelo mais utilizado de ensino, devido ao seu baixo custo de implantação e relativa eficácia, o método tradicional de ensino apresenta diversas falhas e limitações, dentre as quais podemos citar, falta de dinamismo, baixa motivação dos alunos, baixo incentivo à criatividade, dentre outros. É válido notar, no entanto, que o método tradicional de ensino apresenta vantagens e aplicabilidade em diversos contextos (SHOVEIN *et al.*, 2005).

Figura 2 – Representação do ensino tradicional

Adaptado de
<https://larrycuban.wordpress.com/2012/11/28/traditional-an-unpleasant-word-to-reformers/>

2.5 Metodologia Ativa de Aprendizado

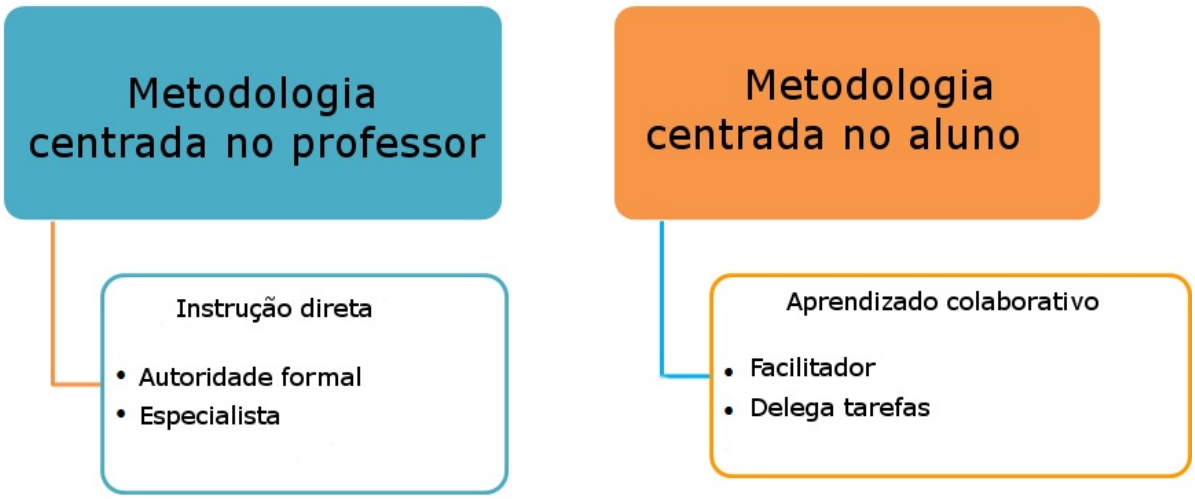
A aprendizagem ativa é um termo cunhado por Reginald William Revans, que pode ser definido como um modelo no qual o estudante está amplamente envolvido no processo de aprendizagem, de forma em que ele busca o próprio conhecimento e não meramente absorve o que é oferecido (PARK; CHOI, 2014). Complementarmente, Faust e Paulson (1998) afirmam que a aprendizagem ativa é, em suma, qualquer atividade de aprendizado exercida por alunos em uma sala de aula, além de ouvir passivamente à lição provinda de um instrutor.

Chickering e Gamson (1999), após realizarem um amplo estudo sobre a realidade de ensino nos Estados Unidos, afirmam que o aprendizado não é uma função de espectador e que os estudantes não têm ganhos significantes, em critérios de conhecimento, ao simplesmente ouvirem seus professores, memorizarem as tarefas dadas e reproduzirem o conteúdo adquirido em testes e avaliações. Para estes autores, os estudantes devem discutir, escrever e praticar o que foi ensinado de forma que o conteúdo aprendido torne-se parte deles mesmos. Ainda, os autores definem como atividades fundamentais do aprendizado ativo exercícios estruturados, discussões desafiadoras, projetos de grupo e *Peer critique* (crítica construtiva mútua).

Bonwell e Eison (1991) afirmam que o modelo tradicional de ensino trata-se de uma abordagem centrada no professor, enquanto os métodos ativos têm seu foco no estudante. A Figura 3 ilustra o papel do educador em cada um dos modelos.

Pode-se perceber que o método tradicional de ensino, centrado no professor, é caracteri-

Figura 3 – Papel do educador nos modelos tradicional e ativo



Adaptado de:
<http://jeancotamora.blogspot.com.br/2015/09/lesson-12-information-technology-in.html>

zado pela instrução direta, uma forma de comunicação unidirecional com origem no professor e com o estudante como alvo. Neste modelo, o educador possui uma autoridade formal, um controle explícito sobre o ambiente de ensino (sala de aula), constitui em um especialista, detentor do conhecimento, cuja finalidade é transmitir o conteúdo estruturado (MIZUKAMI, 1986).

No método de aprendizado ativo, como um modelo centrado no aluno, a função do professor é atuar como facilitador do processo, oferecendo recursos, auxílio e críticas construtivas. Nesta modalidade, o educador não oferece o conteúdo pronto, mas delega tarefas que devem ser realizadas pelos estudantes no intuito de construir o conhecimento dos mesmos. Neste modelo, a forma de instrução é caracterizada pelo aprendizado colaborativo, no qual o coletivo é responsável pela composição do conhecimento, não sendo mais um papel restrito unicamente ao professor (PARK; CHOI, 2014).

A Figura 4 representa um sala de aula da universidade de Swarthmore, Estados Unidos, na qual pratica-se a metodologia de aprendizagem ativa. Nesta imagem é possível perceber elementos comuns ao modelo em questão, tais como a organização da classe em grupos, discussão ativa entre os estudantes, professor como facilitador do processo.

O modelo de aprendizado ativo apresenta resultados, considerando desempenho e receptividade dos estudantes, relevantes em diversas áreas do conhecimento tais como ciências sociais (MCCARTHY; ANDERSON, 2000), linguística (BONWELL; EISON, 1991), direito (BOYLE, 2003), ciências exatas (FREEMAN *et al.*, 2014), ciências da computação (MCCONNELL, 1996) e (SIMON *et al.*, 2004), disciplinas de programação sequencial (KAILA, 2018) e no contexto de computação de alto desempenho (MOORE; DUNLOP, 2016) e (CUENCA; GIMéNEZ, 2016).

Figura 4 – Representação do aprendizado ativo

Fonte: <https://blogs.swarthmore.edu/its/2017/04/28/active-learning-inside-the-classroom/>

2.6 Técnicas de Aprendizado Ativo

Considerando a hierarquia proposta por ANTHONY (1963), discutida na Seção 2.3, define-se como técnica de aprendizado ativo a estratégia, composta pelo conjunto de atividades e mecanismos pontuais que permitem a aplicação dos fundamentos previstos pela metodologia de aprendizado ativo. Em outras palavras, técnica, no contexto deste trabalho, trata-se do procedimento pragmático que viabiliza os preceitos da metodologia em questão, sendo, efetivamente, uma especificação da mesma.

Bonwell e Eison (1991) classificam as técnicas de aprendizado ativo, primariamente, em função de sua orientação ou aspecto, obtendo-se as seguintes categorias: técnicas baseadas em resolução de problemas e técnicas de aprendizado cooperativo.

Os meios de aplicar o aprendizado ativo não são mandatoriamente restritos à classificações e, tampouco, requerem formalizações cerceadoras. Qualquer técnica ou mecanismo, desde que fundamentada nos preceitos do método, são satisfatoriamente capazes de garantir o aprendizado ativo, tal como pode ser visto na publicação de Silberman (1996), que propõe um enorme conjunto de técnicas objetivas e simplificadas que permitem a aplicação do método em questão.

A classificação apresentada a seguir tem como finalidade facilitar o agrupamento de técnicas similares e garantir a indexação de elementos com características em comum.

2.6.1 Técnicas Baseadas em Resolução de Problemas

As técnicas de aprendizado ativo baseadas na resolução de problemas tratam-se de um conjunto de atividades sumarizadas na proposição e resolução de um ou mais problemas, cuja solução deve ser provida pelos alunos, com a mínima interferência do professor (BONWELL; EISON, 1991).

Dewey (1924) define quatro etapas básicas para o processo de aprendizado baseado na resolução de problemas, que são:

1. definição de um problema;
2. diagnóstico de possíveis motivos para o problema;
3. busca por soluções alternativas;
4. avaliação das alternativas e verificação da resolução mais apropriadas

Podemos destacar as técnicas de aprendizado baseado em problemas e aprendizado baseado em projetos como as duas principais estratégias no contexto em questão. É válido notar que o aprendizado baseado em problemas, do inglês *Problem Based Learning*, é um exemplo da categoria de mesmo nome, de acordo com a classificação proposta por Bonwell e Eison (1991). Também é pertinente destacar que a sigla, em inglês, para ambas técnicas é a mesma (PBL).

O ***Problem Based Learning*** **(PBL)**, Aprendizado Baseado em Problemas, é uma técnica de ensino em que problemas do mundo real, usualmente complexos, são usados como veículo para promover o aprendizado dos alunos sobre conceitos e princípios, em oposição à apresentação direta de fatos e conceitos. Além do conteúdo do curso, o PBL pode promover o desenvolvimento de habilidades de pensamento crítico, resolução de problemas e comunicação. Também pode proporcionar oportunidades para exercitar a capacidade de trabalho em grupo e realização de pesquisa científica (DUCH; GROH; ALLEN, 2001).

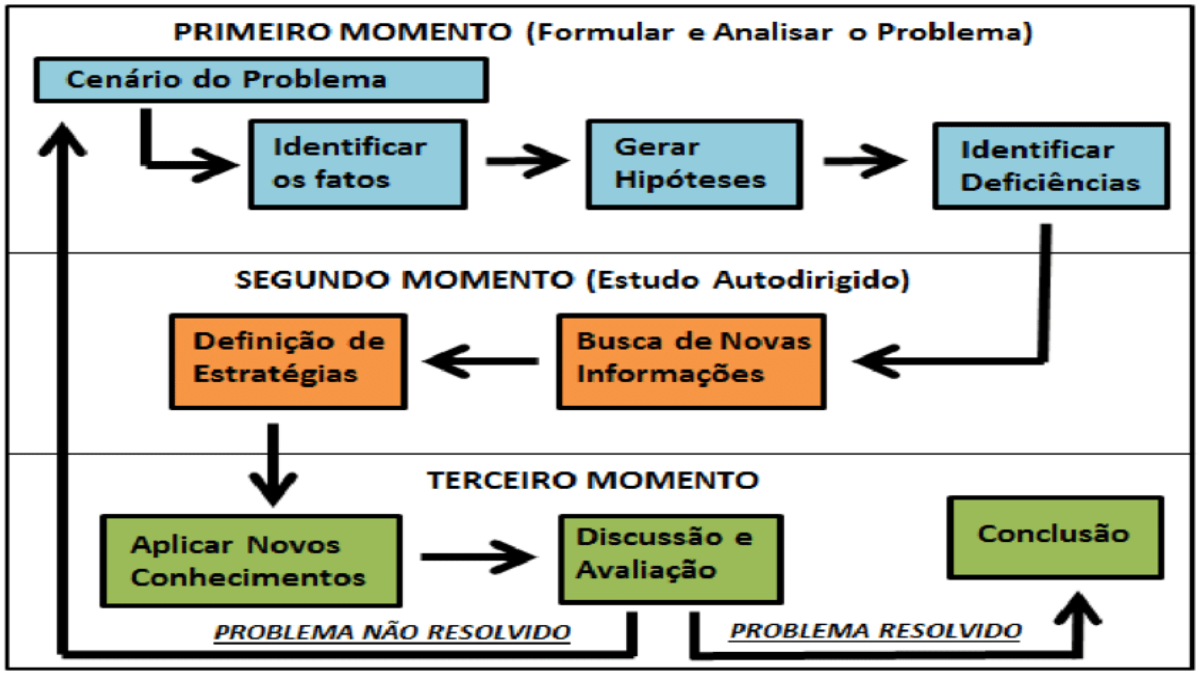
Duch, Groh e Allen (2001) destacam as características fundamentais dos problemas utilizados no PBL, listadas a seguir:

- o problema deve motivar os alunos a compreender profundamente os conceitos;
- o problema deve exigir que os alunos tomem decisões fundamentadas e defendam-nas;
- o problema deve incorporar os objetivos do conteúdo de forma a conectá-lo às disciplinas e conhecimentos anteriores;
- se utilizado para um trabalho em grupo, o problema necessita de um nível de complexidade alto o suficiente para assegurar que os alunos trabalhem juntos para resolvê-lo;

- se utilizado para um trabalho com vários estágios, as etapas iniciais do problema devem ser envolventes e atrativas, com um nível de complexidade inferior às etapas mais avançadas, no intuito de motivar os estudantes a desenvolverem todas as atividades do roteiro proposto.

A Figura 5 ilustra as etapas do aprendizado baseado em problemas.

Figura 5 – O processo de aprendizado baseado em problemas.



Fonte: Lopes (2017).

A referida ilustração indica a divisão das etapas do aprendizado baseado em problemas, fundamentada no algoritmo proposto por Dewey (1924), apresentado no início desta Seção. No primeiro momento, ocorre a análise inicial do problema em questão, na qual identificam-se suas características e propõem-se soluções iniciais, baseadas nas hipóteses primariamente levantadas. Em um segundo momento, ocorre um refinamento da solução proposta inicialmente, no qual novas informações são acrescentadas e identificam-se as estratégias necessárias para resolução do problema. No terceiro momento ocorre, efetivamente, a resolução do problema, a partir da aplicação dos conhecimentos necessários e avaliação da solução proposta, que gera uma resposta final para o referido problema.

O ***Project Based Learning*** **(PBL)**, Aprendizado Baseado em Projetos, é uma técnica de aprendizado baseada na dinamização de sala de aula, na qual acredita-se que os estudantes adquiram conhecimentos aprofundados e sólidos sobre um conteúdo a partir da resolução de desafios baseados em situações do mundo real (DEWEY, 1924). Os alunos permanecem tempo considerável investigando e trabalhando na solução de uma ou um conjunto de questões, cuja complexidade é desafiadora e justifica os requisitos de tempo. Trata-se de um estilo de aprendizado ativo e baseado em inquérito (*inquiry based*) (MARKHAM, 2011).

A Figura 6 ilustra os elementos principais que definem o aprendizado baseado em projetos.

Figura 6 – Características da Aprendizagem Baseada em Projetos

Fonte: <https://blogs.gazetaonline.com.br/conexaodigital/1624/conheca-a-aprendizagem-baseada-em-projetos/>

Conforme apresentado pela figura anterior, a técnica de aprendizado baseado em projetos trata-se de um modelo focado no aluno, no qual acredita-se que o mesmo é capaz de construir o próprio conhecimento, por meio da resolução dos desafios propostos. O professor, neste contexto, atua como facilitador do aprendizado, oferecendo auxílio para o desenvolvimento da solução. Os conteúdos apresentados não são isolados, podendo ser integrados com disciplinas anteriores ou até mesmo combinados com tópicos de diferentes disciplinas. Finalmente, o aprendizado é baseado na aplicação do conhecimento prévio dos estudantes, combinado com as habilidades desenvolvidas durante o curso, no intuito de solucionar problemas reais e, efetivamente, capacitar os estudantes a atuar nestas condições autonomamente (LARMER; MERGENDOLLER, 2012).

As características do aprendizado baseado em problemas e do aprendizado baseado em projetos são, essencialmente, bastante similares, possuem origem em comum e podem ser enquadradas nas mesmas categorias. Larmer, Mergendoller e Boss (2015) definem, pontualmente, as similaridades e diferenças entre os dois métodos. Apresentam-se, primeiramente, as semelhanças entre as técnicas, através do Quadro 1.

As diferenças entre as duas técnicas podem ser vistas por meio do Quadro 2.

Quadro 1 – Similaridades entre aprendizado baseado em problemas e em projetos

Similaridades entre aprendizado baseado em problemas e em projetos:
• Foco em questões ou tarefas não objetivas; • proveem aplicações autênticas de habilidades e conhecimentos; • ênfase na independência e atitude indagativa do estudante; • são mais longas e multifacetadas do que lições e atividades tradicionais.

Adaptado de <https://www.edutopia.org/blog/pbl-vs-pbl-vs-xbl-john-larmer>

Quadro 2 – Diferenças entre aprendizado baseado em problemas e em projetos

Diferenças entre aprendizado baseado em problemas e em projetos:	
Aprendizado baseado em projetos	**Aprendizado baseado em problemas**
Trata de múltiplos assuntos.	Geralmente trata um único assunto.
Dura semanas a meses.	Geralmente mais curto.
Segue etapas mais genéricas.	Segue etapas bem definidas e estruturadas.
Inclui a criação de um produto.	O produto pode ser tangível ou simplesmente uma solução expressa por meio de um documento ou apresentação.
Geralmente trata de problemas reais, com características e tarefas autênticas.	Geralmente utiliza estudos de caso, baseados em problemas reais.

Adaptado de <https://www.edutopia.org/blog/pbl-vs-pbl-vs-xbl-john-larmer>

2.6.2 Técnicas de Aprendizado Colaborativo

O aprendizado colaborativo é um termo abrangente para uma variedade de modelos educacionais que envolvem o esforço intelectual conjunto de estudantes ou alunos e professores. O aprendizado colaborativo refere-se a metodologias e ambientes nos quais os alunos se envolvem em uma tarefa comum, em que cada indivíduo depende e é responsável um pelo outro. Envolve o uso de pequenos grupos para que todos os alunos possam maximizar sua aprendizagem e a de seus pares. É um processo de criação compartilhada: dois ou mais indivíduos interagindo para criar uma compreensão compartilhada de um conceito, disciplina ou área de prática que nenhum tinha anteriormente possuído ou poderia ter vindo por conta própria. As atividades de aprendizagem colaborativa podem incluir escrita colaborativa, projetos grupais e outras atividades (BRUFFEE, 1999).

Individualmente, o termo aprendizagem colaborativa já designa uma técnica de aprendizado ativo, caracterizada pela criação de equipes, atribuição de tarefas que requerem o esforço do coletivo para serem realizadas e construção do conhecimento pela equipe, de forma em que o conteúdo e habilidades são desenvolvidos pela soma das contribuições dos indivíduos e, portanto, a absorção do conhecimento e mútua e multidirecional (DILLENBOURG, 1999).

Exemplos de estratégias de aprendizagem colaborativa tratam-se do aprendizado por

pares (*peer learning*) e aprendizado baseado em equipes, do inglês *Team Based Learning*.

O ***peer learning***, aprendizagem por pares, é uma técnica baseada na troca de experiências entre os próprios estudantes, supervisionada pelo professor, na qual constrói-se o conhecimento em função das soma das habilidades da dupla. O ensino de pares envolve um ou mais estudantes que ensinam outros alunos em uma área de assunto particular e baseia-se na crença de que "ensinar é aprender duas vezes" (GOODLAD; HIRST, 1989).

Define-se a aprendizagem por pares em seu sentido mais amplo, então, como alunos aprendendo uns com os outros de maneira formal e informal. A ênfase é no processo de aprendizagem, adicionalmente inclui-se o suporte emocional que os alunos podem oferecer aos seus semelhantes. Os docentes podem estar ativamente envolvidos como facilitadores do grupo ou podem simplesmente iniciar e supervisionar atividades dirigidas por alunos, como *workshops* ou parcerias de aprendizagem (BOUD; COHEN; SAMPSON, 2014).

O ***Team Based Learning***, da sigla TBL, ou Aprendizado Baseado em Equipes é uma estratégia de aprendizagem colaborativa baseada em evidências, projetada em torno de unidades de instrução, conhecidas como módulos, que são ensinadas em um ciclo de três etapas: preparação, testes de garantia de preparação em classe e exercícios focados em aplicações. Um conteúdo normalmente inclui um módulo (HILLS, 2001).

Michaelsen e Sweet (2011) definem os quatro elementos práticos do TBL, listados a seguir:

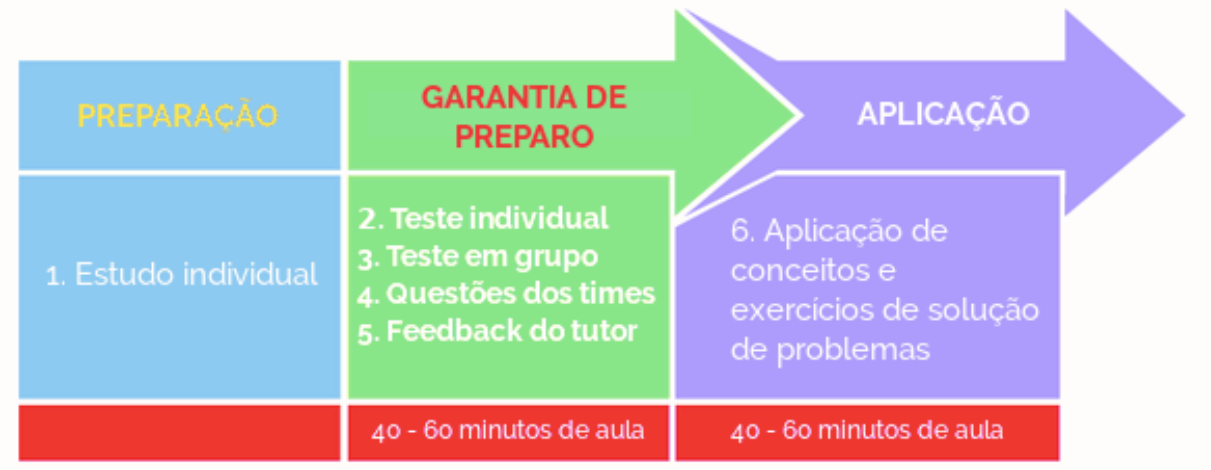
1. Grupos estrategicamente formados e permanentes;
2. Garantia de preparação;
3. Atividades de aplicação que promovem o pensamento crítico e o desenvolvimento em equipe;
4. Avaliação por pares.

A Figura 7 representa as etapas características do TBL, seus componentes e uma estimativa do tempo necessário em sala de aula para realização das atividades do modelo.

Podemos ainda citar, como técnicas de aprendizado ativo, as estratégias de sala de aula invertida e ludificação ou gameficação.

Flipped Classroom ou sala de aula invertida é uma técnica pedagógica híbrida, por combinar elementos de diferentes métodos educacionais, caracterizada pela realização de tarefas de aprendizado fora do contexto de sala de aula. Neste modelo, os estudantes desenvolvem atividades e realizam o estudo do material didático em casa ou outro ambiente e utilizam a sala de aula para discutir o que foi visto, com supervisão do professor (LAGE; PLATT; TREGLIA, 2000).

Figura 7 – Etapas do TBL



Adaptado de Inuwa (2012)

A estratégia em questão não é classificada, em geral, como técnica de aprendizado ativo, embora compartilhe de características similares, como a atuação ativa do estudante na construção do conhecimento e papel do docente como facilitador do aprendizado.

Recentemente, a técnica de sala de aula invertida tem tido resultados relevantes, no aspecto de qualidade do aprendizado, o que motiva aplicações em variadas áreas do conhecimento (BISHOP; VERLEGER, 2013).

Gamefication, gameficação ou ludificação constitui no uso de jogos para enriquecimento de contextos não associados ao desenvolvimento e prática de *games*, tais como no ambiente empresarial ou educacional (HUOTARI; HAMARI, 2012).

No contexto pedagógico, encenações (*roleplay*), simulações e jogos podem ser usados para permitir que os estudantes experimentem situações estressantes, desconhecidas, complexas ou controversas, criando certas circunstâncias que são temporariamente reais, permitindo que os alunos desenvolvam e pratiquem as habilidades necessárias para enfrentar desafios. Os estudantes trabalham em grupos, geralmente apresentam alto nível de motivação e entusiasmo e desenvolvem capacidades individuais, tais como iniciativa e liderança (BONWELL; EISON, 1991).

2.7 Desafios de Programação

Desafio de programação, do inglês *programming challenges*, pode ser definido como uma atividade prática de programação de computadores, usualmente baseada em problemas reais, cuja complexidade é consideravelmente maior do que um mero exercício ou tarefa de programação, justificando, portanto, o termo “desafio”. Para resolução deste tipo de problema é necessário que o estudante tenha conhecimento prévio sobre linguagem e lógica de programação e sobre o contexto tratado no referido desafio. É esperado, também, que ao construir a solução, o

estudante exercite habilidades como criatividade, raciocínio lógico, tomada de decisão, dentre outras. Os desafios de programação não são necessariamente aplicados a uma equipe e nem requerem o ambiente de sala de aula/laboratório de ensino para serem realizados. Desta forma, os estudantes podem participar, individualmente, pela internet, por meio de sistemas automatizados de julgamento/correção (*online judges*) (SKIENA; REVILLA, 2006).

Maratona ou competição de programação, do inglês *programming contest*, refere-se ao esporte intelectual, geralmente organizado em formato de concurso, que prevê a resolução de um conjunto de desafios de programação por uma equipe, durante um período limitado de tempo. As equipes participantes têm o objetivo de solucionar corretamente a maior quantidade de problemas, no menor tempo possível, sendo pontuadas pela combinação do número de problemas resolvidos e tempo gasto. Usualmente, os participantes são motivados por prêmios em dinheiro, recomendações profissionais e pelo próprio reconhecimento de sua habilidade em programação. Uma maratona de programação pode ser realizada tanto presencialmente, quanto remotamente (KHERA; ASTRACHAN; KOTZ, 1993).

Podemos citar, como exemplos de maratonas de programação relevantes, a maratona nacional da Sociedade Brasileira de Computação (<http://maratona.ime.usp.br/>), cujos vencedores garantem vaga para a *ACM International Collegiate Programming Contest* (ICPC) (<https://icpc.baylor.edu/>), organizada pela ACM e considerado o evento mais relevante neste contexto. Além disso, há a Olimpíada Brasileira de Informática (OBI) (<https://olimpiada.ic.unicamp.br/>), visando despertar o interesse na área de ciência da computação e promover competições para alunos inciciante em computação. Outro evento importante deste cenário trata-se das Olimpíadas Internacionais de Informática (IOI) (<http://www.ioinformatics.org/index.shtml>) (SKIENA; REVILLA, 2006).

No contexto de programação paralela, existe o *Concurso de Programación Paralela* (<http://luna.inf.um.es/2017/>), realizado desde 2011 na Espanha (ALMEIDA *et al.*, 2012). No Brasil, ocorrem maratonas de programação paralela no Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD) (<http://lspd.mackenzie.br/marathon/current/index.pt.html>), Escola Regional de Alto Desempenho de São Paulo (ERAD-SP) (<http://eradsp2018.lsc.ic.unicamp.br/>) e Escola Regional de Alto Desempenho do Rio Grande do Sul (ERAD-RS) <http://www.inf.ufrgs.br/erad2018/maratona.html>.

Existe também competições denominadas *hackhaton*. Um *hackathon* (também conhecido como *hack day*, *hackfest* ou *codefest*) é um evento competitivo de desenvolvimento de *software*, em que os programadores e outros envolvidos em implementação de aplicações, incluindo *designers* gráficos, *designers* de interface, gerentes de projeto e outros, muitas vezes incluindo especialistas, que colaboram intensamente em projetos de *software*. O objetivo de um *hackathon* é criar software utilizável. *Hackathons* tendem a ter um foco específico, que pode incluir a linguagem de programação utilizada, o sistema operacional, uma aplicação, uma API, redes de computadores, criptografia ou outros assuntos de interesse para a comunidade de computação. Em

outros casos, não há restrição no tipo de software que está sendo criado (BRISCOE; MULLIGAN, 2014).

Os *hackthons* se diferenciam das maratonas de programação regulares por não terem foco na resolução de problemas ou desafios, e sim na produção de *software* funcional e completo dentro das temáticas propostas pelo evento (BRISCOE; MULLIGAN, 2014).

Dentre os sistemas automatizados de julgamento, que são *web sites* especializados na promoção e avaliação de desafios de programação, podemos citar o UVA (<https://uva.onlinejudge.org/>), o SPOJ <http://www.spoj.com/> e Code Chef (<https://www.codechef.com/>). Estes sistemas disponibilizam problemas utilizados em maratonas de programação e interfaces para submissão e avaliação automática da corretude das soluções desenvolvidas.

Desafios ou maratonas de programação, quando aplicados no contexto pedagógico, são capazes de proporcionar um ambiente de ensino-aprendizado mais dinâmico e mais atrativo para os estudantes. Tais recursos favorecem o desenvolvimento de habilidades tais como capacidade de trabalhar em equipe e sob pressão, criatividade, capacidade de resolução de problemas, dentre outros. Favorece a aplicação dos conceitos teóricos e o exercício das habilidades de projeto e desenvolvimento de algoritmos. A recepção dos alunos e desempenho na resolução dos problemas são consideravelmente favorecidos por estes recursos (KHERA; ASTRACHAN; KOTZ, 1993).

Há um série de referências literárias que atestam a validade da utilização de desafios ou maratonas de programação, como recurso educacional aplicado ao aprendizado de disciplinas de programação. Dentre os referidos estudos podemos citar, em ordem cronológica, as publicações de Khera, Astrachan e Kotz (1993), Shilov e Yi (2002), Boersen e Phillipps (2006), Manev, Kelevedjiev e Kapralov (2007) e Combéfis e Wautelet (2014).

No contexto do presente trabalho, existem publicações que comprovam a eficácia do método, também, para o aprendizado de programação paralela. As publicações de Almeida *et al.* (2012) e Giménez (2016) descrevem a utilização de maratonas e desafios de programação de forma bastante similar à praticada neste trabalho, sendo a última referência uma evidência do sucesso da utilização de aprendizado baseado em problemas, uma técnica de aprendizado ativo, no contexto em questão. As demais características destes trabalhos, similaridades e diferenças em relação ao presente, serão elaboradas posteriormente.

É razoável afirmar, com base nas referências apresentadas anteriormente, que a estratégia de maratonas ou desafios de programação constitui em um valioso recurso de apoio ao aprendizado, possuindo resultados consideráveis e relevantes, inclusive no contexto específico do presente trabalho: a disciplina de programação paralela.

2.7.1 Sistemas de Apoio à Maratona de Programação

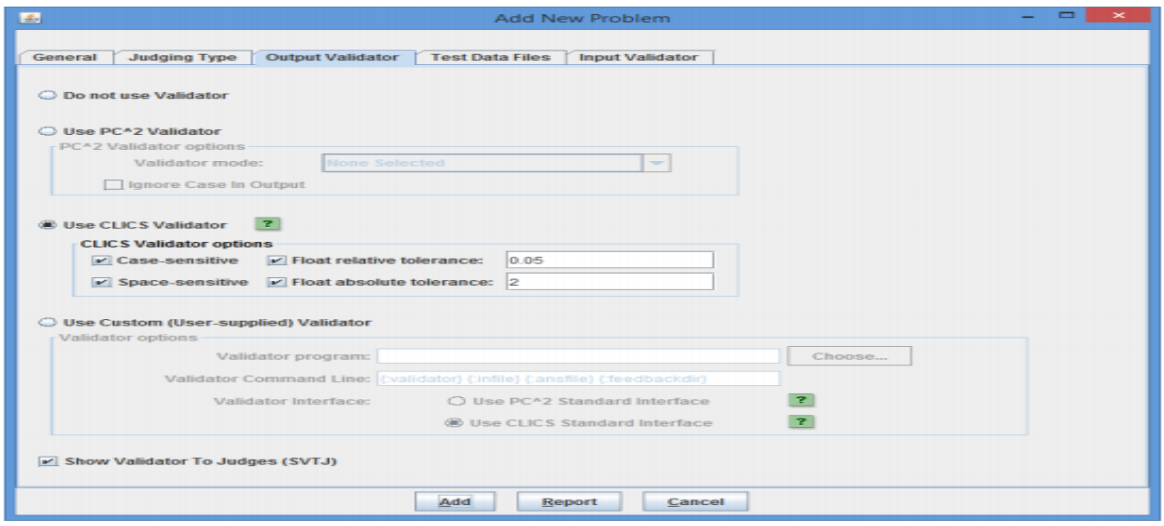
Para a realização de uma maratona de programação, ainda que no contexto educacional e não competitivo, é necessário, minimamente, a elaboração dos desafios que constituem o concurso e um sistema de apoio ou gerenciamento para o referido evento. O dito *software* deve ser capaz de permitir o cadastro dos problemas e suas descrições, aceitar múltiplas submissões simultâneas de respostas para os problemas, oferecer um mecanismo de classificação/pontuação para as equipes e permitir o julgamento ou avaliação das soluções submetidas (KHERA; ASTRACHAN; KOTZ, 1993).

Existem diversos sistemas capazes de satisfazer os requisitos mencionados no parágrafo anterior, cada qual com suas características e funcionalidades particulares. Dentre esse conjunto de programas, são particularmente relevantes o PC^2, utilizado no *ACM International Collegiate Programming Contest*, o BOCA, amplamente utilizado no Brasil (FERREIRA, 2004) e o *Mooshak*, desenvolvido pela Universidade do Porto e utilizado na Europa (LEAL; SILVA, 2003).

O **PC^2** é um *software* projetado para permitir a gestão de competições de programação em uma variedade de ambientes de computação. O PC2 permite aos concorrentes (equipes) enviarem programas em uma rede para avaliação de juízes. Os juízes podem recompilar o programa enviado, executá-lo, visualizar o código fonte e / ou resultados de execução, e enviar uma resposta de volta ao time. O sistema também suporta um modo de "julgamento automatizado" no qual o julgamento é realizado por software e não por juízes humanos (ACM, 2015).

A Figura 8 ilustra a interface do PC^2, referente ao módulo de inserção de problemas.

Figura 8 – Interface gráfica do PC^2



Fonte: <http://pc2.ecs.csus.edu/doc/v9/9.5.3/WhatsNew.pdf>

O sistema registra automaticamente os horários e os arquivos enviados, mantém e exibe a classificação atual do concurso de várias maneiras, e permite que os juízes recuperem e reproduzam as execuções arquivadas. Ele também fornece um mecanismo para os concorrentes enviar

pedidos de esclarecimento e consultas aos juízes, e para que os juízes respondam às solicitações e emitam boletins ou relatórios para equipes. Além disso, o PC^2 suporta concursos simultaneamente em várias localizações, transmitindo automaticamente informações permanentes do concurso entre os *sites* e gerando um painel único de classificação para cada ambiente remoto (ACM, 2015).

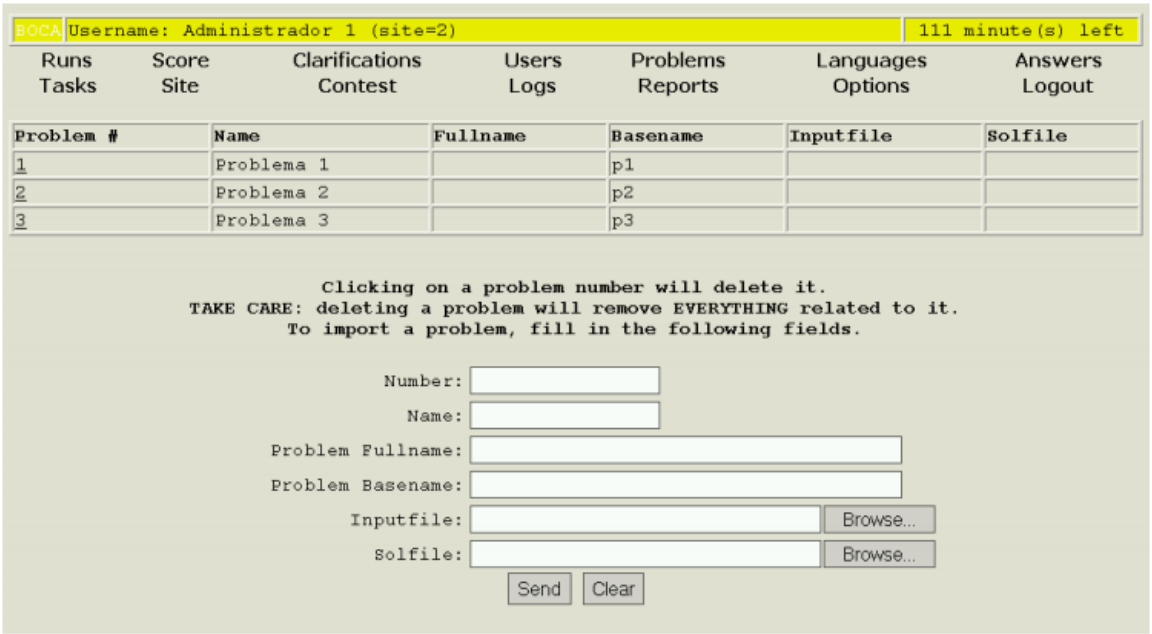
O **BOCA** é um sistema um sistema de apoio à competições de programação desenvolvido para ser utilizado na Maratona de Programação da Sociedade Brasileira de Computação, o que tem acontecido desde 2002. Trata-se de um *software* aberto, robusto, configurável e escalável. O sistema possui uma arquitetura dividida em cinco partes, de acordo com as particularidades e permissões de cada usuário: time, juiz, administrador, *staff* e placar (FERREIRA, 2004).

O BOCA apresenta funcionalidades similares ao PC^2, com exceção da ausência de um módulo de correção automática.

As características relativas à implementação do sistema BOCA e o detalhamento das suas funcionalidades podem ser vistas em Ferreira (2004) e em BOCA (2010).

A Figura 8 ilustra a interface do BOCA, destacando o módulo reservado ao administrador do sistema.

Figura 9 – Interface gráfica do BOCA



Fonte: Ferreira (2004).

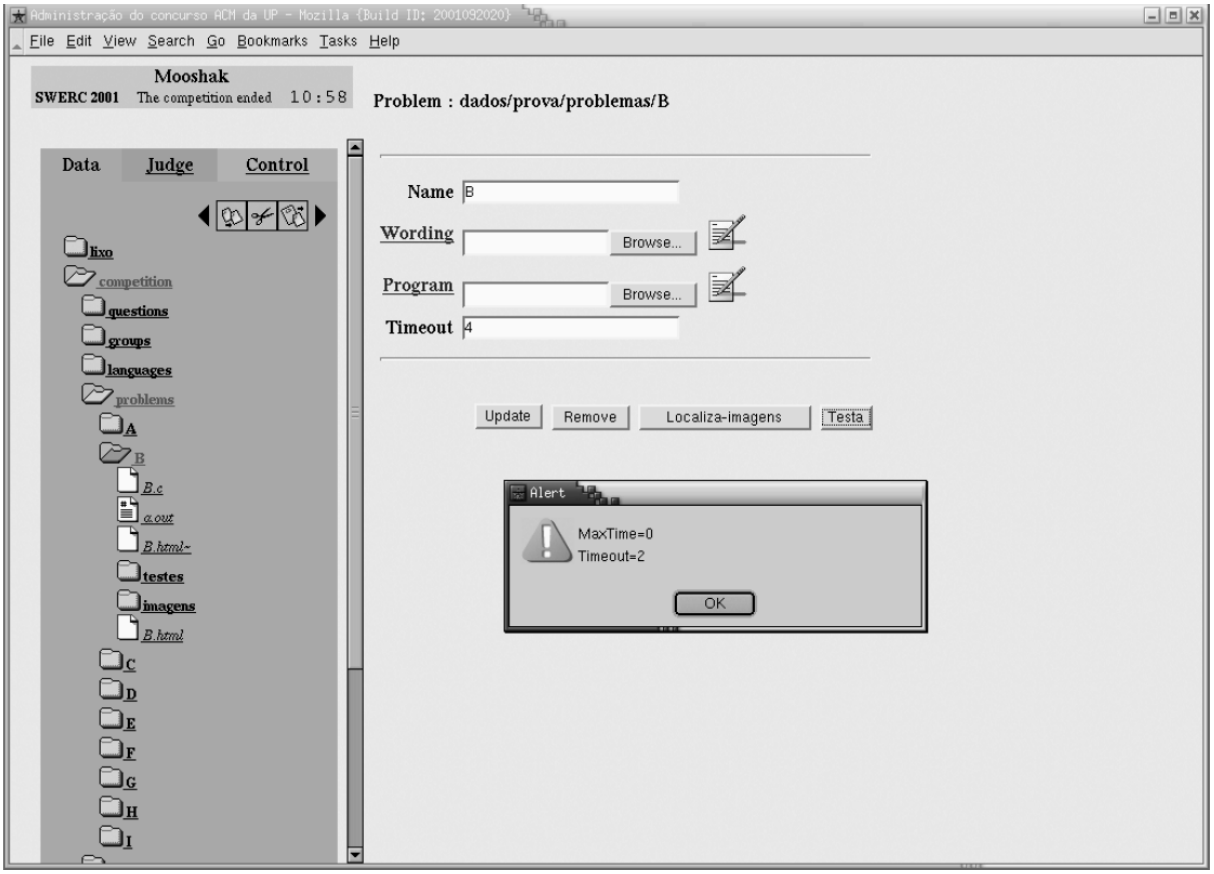
O ***Mooshak*** é um sistema de gestão de maratonas de programação, originalmente desenvolvido pela Universidade do Porto, Portugal. Trata-se de um *software* constituído por uma arquitetura escalável, que pode ser usado em concursos de programação privados, hospedados em um único servidor, e em concursos distribuídos complexos, ou maratonas públicas de pro-

gramação; possui um robusto sistema de gerenciamento de dados que favorece procedimentos simples para armazenamento, replicação, *backup* de dados e recuperação de falhas, utilizando objetos persistentes; possui capacidades de julgamento automático, que podem auxiliar os juízes humanos na avaliação das soluções submetidas pelas equipes; tem medidas de segurança incorporadas para impedir que os usuários interfiram com o progresso regular dos eventos. (LEAL; SILVA, 2003).

O *Mooshak* é um software *open source*, cuja documentação, implementação e detalhamento de suas funcionalidades e arquitetura podem ser vistos em Mooshak (2016) e Leal e Silva (2003).

A Figura 8 ilustra a interface apresentada ao administrador no *Mooshak*.

Figura 10 – Interface gráfica do *Mooshak*



Fonte: Leal e Silva (2003).

O referido sistema apresenta funcionalidades similares às disponíveis no PC2 e no BOCA. Suas vantagens em relação aos demais sistemas são o mecanismo de validação de soluções em tempo real implementado (julgamento automatizado), escalabilidade, disponibilidade e integridade de dados.

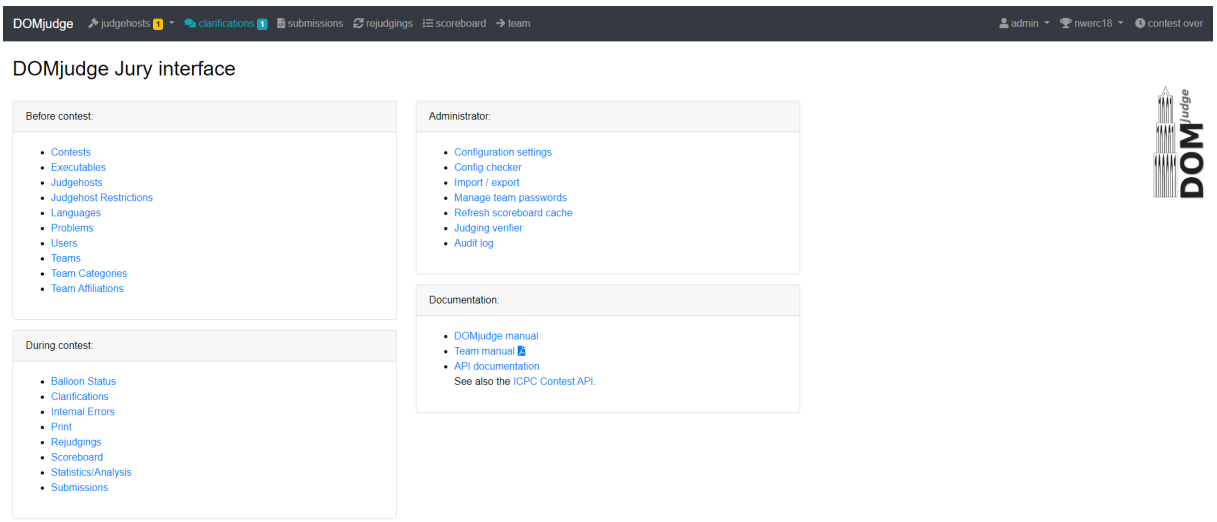
Existe uma versão do *Mooshak* adaptada para a utilização em maratonas de programação paralela, que é utilizada para realização do concurso de programação paralela espanhol (*Concurso*

de Programación Paralela) (Concurso de Programación Paralela, 2017) e, inclusive, já foi utilizado como ferramenta de apoio para ministrar um curso de programação paralela, na Universidade de Murcia, Espanha (GIMÉNEZ, 2016).

Atualmente, o *Mooshak* está em sua versão 2.0 (<https://mooshak2.dcc.fc.up.pt/>) na qual foram inseridas diversas funcionalidades voltadas à usabilidade, módulos adicionais para gamificação e recursos de interface gráfica. A documentação desta versão é bastante reduzida, limitando-se a vídeos demonstrando algumas funcionalidades do sistema. Na presente dissertação, essa versão do *software* foi utilizada e documentos para auxiliar a instalação e implantação da mesma foram desenvolvidos e estão disponíveis publicamente.

O *DOMjudge* é um sistema de apoio à realização de maratonas de programação desenvolvido por Jaap Eldering, Nicky Gerritsen, Keith Johnson, Thijs Kinkhorst, Tobias Werth e outros contribuidores. Este sistema também apresenta sistema de correção automatizada, interfaces gráficas para competidores e juizes e ferramentas para cadastrar problemas e linguagens/modelos de programação, além de uma *API REST* que facilita a integração a outros sistemas. Atualmente, o *DOMjudge* é utilizado no *ICPC* (PHAM; NGUYEN, 2019). A Figura 11 ilustra a interface do juíz no *DOMjudge*.

Figura 11 – Interface gráfica do *DOMjudge*



Fonte: <https://www.domjudge.org/demoweb/jury>

2.8 Considerações Finais

Neste capítulo, foram apresentadas as definições acerca da teoria pedagógica, incluindo a terminologia praticada ao longo do texto, características da metodologia de ensino ativo e suas principais técnicas. Definiu-se metodologia como o conjunto de práticas que permite descrever o ensino ou aprendizado de um conteúdo.

Destacou-se o método de aprendizado ativo em função das publicações relevantes, cujo resultado demonstra a viabilidade da estratégia.

Foram evidenciadas as técnicas de aprendizado baseado na resolução de problemas, devido à sua aplicabilidade no contexto do contexto de disciplinas de programação.

Descreveram-se desafios e maratonas de programação, como estratégias para o desenvolvimento de capacidades individuais e coletivas, além do exercício do conteúdo prático do curso de programação paralela pelos estudantes. Finalmente, apresentaram-se os aspectos dos sistemas de apoio à realização de maratonas de programação, um recurso educacional valioso para interação e avaliação dos estudantes.

Descreveu-se, em suma, por meio de uma abordagem *top down*, neste capítulo, no qual está inserido o presente trabalho e justificativas para seu desenvolvimento.

CAPÍTULO

3

ENSINO DE PROGRAMAÇÃO PARALELA

3.1 Considerações Iniciais

Este Capítulo apresenta, inicialmente, os conceitos essenciais para o ensino de computação paralela, sob a ótica do ACM/IEEE-CS Joint Task Force on Computing Curricula (2013), considerando as ementas, recursos, carga horária e demais particularidades das disciplinas que fundamentam o referido conceito, nas principais universidades do mundo.

Posteriormente, apresenta-se as principais e mais recentes estratégias utilizadas para apoio do ensino e aprendizado da disciplina em questão.

Por fim, estabelecem-se as lacunas do processo de ensino-aprendizagem ainda existentes, considerando o que foi observado na literatura, e que, por consequência justificam novas abordagens que sejam capazes de solucionar tais problemas e promover um ensino mais eficaz, no contexto das disciplinas em questão.

3.2 Conceitos Essenciais

Embora frações de seu conteúdo já sejam ministrados em universidades desde a década de 1980, a disciplina de programação paralela nem sempre foi considerada um componente fundamental e obrigatório nas grades curriculares de computação. Antes de ser considerada uma área essencial de estudo, foram feitas diversas iniciativas, como a descrita em Ernst e Stevenson (2008), que buscavam implantar tópicos de programação paralela em disciplinas tidas como imprescindíveis em grades curriculares de cursos de computação. Foram propostas abordagens que não tinham como objetivo ensinar programação paralela como uma nova área de estudo, mas que integrassem conceitos de paralelismo em disciplinas-base de um curso de graduação em computação (ERNST; STEVENSON, 2008). No entanto, devido à crescente importância da programação paralela, atribuída, primariamente, à necessidade de implementação

de sistemas complexos, redução de tempo de resposta em aplicações críticas e pela expansão da computação de alto desempenho; em 2013, o modelo curricular apresentado em ACM/IEEE-CS Joint Task Force on Computing Curricula (2013), chancelado pela ACM juntamente com a IEEE e elaborado por um grupo de especialistas das referidas instituições, propôs a área de Computação Paralela e Distribuída como essencial para a formação de alunos em ciência da computação, definindo tópicos fundamentais e como a aprendizagem deve ser avaliada.

Para o presente projeto, serão considerados como conceitos essenciais no ensino de programação paralela os tópicos que constituem a ementa referente à cada subárea de Computação Paralela e Distribuída, previstos em ACM/IEEE-CS Joint Task Force on Computing Curricula (2013). As referidas subáreas são apresentados a seguir, em mesma ordem daquela praticada no documento: Fundamentos de Paralelismo, Decomposição Paralela, Comunicação e Coordenação, Algoritmos Paralelos, Análise e Programação, Arquitetura Paralela, Desempenho Paralelo, Sistemas Distribuídos, Computação em Nuvem e Modelos Formais e Semântica.

Cada componente constituinte da área de conhecimento do currículo é classificado em "fundamental" (*Core*) ou "optativo" (*Elective*), com a divisão daqueles considerados fundamentais em duas camadas: "Camada Principal 1" (*Tier-1*) e "Camada Principal 2" (*Tier-2*).

Resumidamente e em sequência, apresenta-se os tópicos de aprendizado, extraídos de ACM/IEEE-CS Joint Task Force on Computing Curricula (2013), considerando apenas o conteúdo de computação paralela. Os conteúdos de Sistemas Distribuídos, Computação em Nuvem e Modelos Formais e Semântica não serão detalhados aqui por não constituírem no foco do presente trabalho.

Tópicos de Aprendizagem em Computação Paralela

I) Fundamentos de Paralelismo

Camada Principal 1

- Múltiplas computações simultâneas;
- Paralelismo versus concorrência;
- Paralelismo, comunicação e coordenação;
- Erros de programação não encontrados em programação sequencial.

II) Decomposição Paralela

Camada Principal 1

- Comunicação, coordenação e sincronização;
- Independência e particionamento;

Camada Principal 2

- Conceitos básicos de decomposição paralela;
- Decomposição por tarefas;
- Decomposição por dados.

III) Comunicação e Coordenação

Camada Principal 1

- Memoria compartilhada;
- Consistência e como ela garante programas livres de condições de corrida;

Camada Principal 2

- Passagem de mensagem;
- Atomicidade.

Optativos

- Consenso;
- Ações condicionais.

IV) Algoritmos paralelos, análise e programação

Camada Principal 2

- Caminhos críticos, tarefas e extensão e lei de Amdahl;
- *Speedup* e escalabilidade;
- Algoritmos naturalmente paralelos;
- Classes de algorítmicos paralelos.

Optativos

- Algoritmos paralelos baseados em grafos;
- Computações paralelas em matrizes;
- Produtor-consumidor e algoritmos com *pipeline*;
- Exemplos de algoritmos paralelos não escaláveis.

V) Arquitetura Paralela

Camada Principal 1

- Processadores *multicore*;
- Memória compartilhada versus memória distribuída;

Camada Principal 2

- Multiprocessamento simétrico (SMP);
- SIMD, processamento vetorial.

Optativos

- GPUs e co-processamento;
- Taxonomia de Flynn;
- Programação paralela em nível de instrução;
- Problemas relacionados à memória;
- Topologias.

VI) Desempenho Paralelo

Optativos

- Balanceamento de carga;
- Mensuração de desempenho;
- Escalonamento e contenção;
- Avaliação do *overhead* de comunicação;
- Gestão de dados;
- Uso e gerenciamento de energia.

Além dos tópicos apresentados em cada área de conhecimento e sua classificação como principal ou optativo, são descritos resultados de aprendizagem primordiais desejáveis visando a compreensão de cada aluno ao estudar os tópicos das subáreas relacionadas. Analisando os resultados de aprendizagem descritos no modelo curricular referenciado, é possível perceber o que é esperado abordar no contexto específico de cada subárea de conhecimento. Além disso, Cada resultado de aprendizagem está associado a um nível de domínio esperado do mesmo. Tais níveis de domínio foram definidos da seguinte maneira por ACM/IEEE-CS Joint Task Force on Computing Curricula (2013):

- **Familiaridade:** O aluno entende o que é ou o que significa um conceito. Provê a resposta para a pergunta “o que você sabe sobre isso?”;

- **Aplicação:** O aluno é capaz de usar ou aplicar um conceito de forma concreta.Provê a resposta para a pergunta “o que você sabe fazer?”;
- **Avaliação:** O aluno consegue analisar um conceito a partir de pontos de vista distintos e justificar a seleção de uma abordagem específica para resolver uma situação. Esse nível de domínio implica em mais do que usar um conceito, envolve a capacidade de selecionar uma abordagem apropriada dentre múltiplas alternativas cabíveis. Provê a resposta para a pergunta “por que você faz isso?”.

Como ilustração para os níveis de domínio descritos previamente, considere o conhecimento de diferentes estruturas de repetição. Ao nível da "Familiaridade", um aluno deve ter uma definição do conceito de iteração no desenvolvimento de *software* e saber a razão pela qual é considerada uma técnica útil. Para mostrar o domínio no nível de "Aplicação", o aluno deve ser capaz de escrever um programa aplicando corretamente uma estrutura de repetição. Compreender em nível de "Avaliação"exige que um aluno compreenda vários métodos de iteração e escolha adequadamente entre eles para solucionar um problema específico.

Na listagem disposta a seguir, são apresentados os resultados de aprendizagem definidos para a área de conhecimento denominada Computação Paralela e Distribuída no modelo curricular descrito em ACM/IEEE-CS Joint Task Force on Computing Curricula (2013). A estrutura destes elementos segue a mesma hierarquia dos tópicos de aprendizado definidos anteriormente.

Resultados de Aprendizagem

I) Fundamentos de Paralelismo

Camada Principal 1

1. Distinguir diferentes métodos que permitam acesso eficiente a um recurso compartilhado. [Familiaridade]
2. Distinguir diferentes métodos de sincronização, suas vantagens e desvantagens. [Familiaridade]
3. Distinguir diferentes causas de corrida de dados [Familiaridade].

II) Decomposição Paralela

Camada Principal 1

1. Explicar porque a sincronização é necessária em um programa paralelo específico. [Aplicação]
2. Identificar oportunidades para particionar um programa serial em módulos paralelos independentes. [Familiaridade]

Camada Principal 2

3. Escrever um algoritmo paralelo correto e escalável. [Aplicação]
4. Paralelizar um algoritmo aplicando a decomposição baseada em tarefas. [Aplicação]
5. Paralelizar um algoritmo aplicando decomposição paralela por dados. [Aplicação]
6. Desenvolver um programa usando atores e/ou processos reativos. [Aplicação]

III) Comunicação e Coordenação

Camada Principal 1

1. Utilizar exclusão mútua para evitar uma determinada condição de corrida. [Aplicação]
2. Fornecer um exemplo de ordenação de acessos entre atividades concorrentes (por exemplo, programa com uma condição de corrida) que não é sequencialmente consistente. [Familiaridade]

Camada Principal 2

3. Fornecer um exemplo de um cenário no qual o envio de uma mensagem bloqueante pode causar *deadlock*. [Aplicação]
4. Explicar quando e por que o *multicast* ou mensagens baseadas em eventos podem ser preferíveis como alternativas a outros métodos. [Familiaridade]
5. Escrever um programa que finaliza sua execução corretamente quando todo um conjunto de tarefas simultâneas foi concluído. [Aplicação]
6. Utilizar uma fila corretamente sincronizada para armazenar dados em *buffer* entre as atividades. [Aplicação]
7. Explicar por que verificação de condições prévias e ações com base nessas verificações devem compartilhar a mesma unidade de atomicidade para ser efetiva. [Familiaridade]
8. Desenvolver um programa de teste que possa revelar erros de programação concorrente. Por exemplo, perder uma atualização quando duas atividades tentam incrementar a mesma variável. [Aplicação]
9. Descrever pelo menos uma técnica de projeto para evitar falhas de execução em programas que usam vários bloqueios ou semáforos. [Familiaridade]
10. Descrever os méritos relativos do controle de concorrência otimista versus conservador, sob diferentes taxas de contenção entre as atualizações. [Familiaridade]
11. Dar um exemplo de um cenário no qual uma tentativa de atualização otimista talvez nunca seja concluída. [Familiaridade]

Optativos

12. Usar semáforos ou variáveis de condição para bloquear segmentos enquanto uma condição prévia necessária seja mantida. [Aplicação]

IV) Algoritmos paralelos, análise e programação

Camada Principal 2

1. Definir caminho crítico, trabalho e *span*. [Familiaridade]
2. Calcular trabalho e *span* e determinar o caminho crítico em relação a um diagrama de execução paralelo. [Aplicação]
3. Definir *speedup* e explicar a noção de escalabilidade de um algoritmo neste este contexto. [Familiaridade]
4. Identificar tarefas independentes em um programa que pode ser paralelizado. [Aplicação]
5. Identificar características de uma carga de trabalho que permite ou impede que seja naturalmente paralelizada. [Familiaridade]
6. Implementar um algoritmo paralelo de divisão e conquista e medir empiricamente seu desempenho em relação ao seu análogo sequencial. [Aplicação]
7. Decompor um problema (por exemplo, contagem do número de ocorrências de alguma palavra em um documento) por meio de operações de mapeamento e redução (*map and reduce*). [Aplicação]

Optativos

8. Fornecer um exemplo de problema que se encaixe no paradigma produtor-consumidor. [Familiaridade]
9. Dar exemplos de problemas em que *pipelining* seria um meio efetivo de paralelização. [Familiaridade]
10. Implementar um algoritmo matricial paralelo. [Aplicação]
11. Identificar os problemas que surgem nos algoritmos produtor-consumidor e mecanismos que possam ser usados para tratá-los. [Familiaridade]

V) Arquitetura Paralela

Camada Principal 1

1. Explicar as diferenças entre memória compartilhada e distribuída. [Familiaridade]

Camada Principal 2

2. Descrever a arquitetura SMP e destacar suas principais características. [Familiaridade]
3. Caracterizar os tipos de tarefas que são melhor tratadas em máquinas SIMD. [Familiaridade]

Optativos

4. Descrever as vantagens e limitações das GPUs em relação às CPUs. [Familiaridade]
5. Explicar as características de cada classificação na taxonomia da *Flynn*. [Familiaridade]
6. Descrever o suporte em nível de *assembly* para operações atômicas. [Familiaridade]
7. Descrever os desafios na manutenção da coerência de cache. [Familiaridade]

VI) Performance Paralela

Optativos

1. Detectar e corrigir um desbalanceamento de carga. [Aplicação]
2. Calcule as implicações da lei de Amdahl para um algoritmo paralelo particular. [Aplicação]
3. Descreva como a distribuição de dados pode afetar os custos de comunicação de um algoritmo. [Familiaridade]
4. Detectar e corrigir uma instância de falso compartilhamento. [Uso]
5. Explicar o impacto de escalonamento no desempenho paralelo. [Familiaridade]
6. Explicar os impactos de desempenho causados pela localidade de dados. [Familiaridade]
7. Explicar o impacto e o *trade-off* relacionados ao uso de energia no desempenho paralelo. [Familiaridade]

Um resultado complementar àquele discutido ao longo desta Seção é a publicação de Prasad *et al.* (2014), cujos resultados estão disponíveis em <https://grid.cs.gsu.edu/~tcpp/curriculum/>. O referido estudo descreve a proposta curricular para o ensino de computação paralela e distribuída elabora por um conjunto de renomados especialistas, apoiada pelas instituições NSF/IEEE e TCPP. Esta publicação embasou a elaboração da proposta de (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013), embora o detalhamento de cada um dos tópicos que compõem a ementa da área de conhecimento é consideravelmente superior no primeiro trabalho. Os conteúdos referentes ao domínio de programação paralela são virtualmente idênticos em ambas as publicações, apresentando diferenças na profundidade com a qual os temas são tratados e a forma em que os mesmos são categorizados.

Por meio da combinação dos tópicos e resultados de aprendizados propostos nas diretrizes curriculares de (ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013) e Prasad *et al.*

(2011), é possível compreender, em extensão e profundidade, o conteúdo atribuído à disciplina de programação paralela. No presente trabalho, o enfoque é atribuído aos aspectos práticos do conteúdo, tais como modelos de programação e a algoritmos paralelos, devido à natureza do projeto. Entretanto, os aspectos teóricos fundamentais também serão tratados, ainda que indiretamente, por meio de práticas derivadas do método de sala de aula invertida.

3.3 Lacunas

O mapeamento sistemático desenvolvido, detalhado no Capítulo 4, é capaz de ilustrar que, embora existam publicações que investiguem e questionem a qualidade do ensino-aprendizado de programação paralela e proponham novas soluções para os problemas identificados, há, ainda, dificuldades e desafios não superados no contexto do ensino e da aprendizagem da programação paralela.

Muresano, Rexachs e Luque (2012) afirmam que a popularização dos computadores pessoais *multi-core*, surgimento das arquiteturas *many-core*, desenvolvimento e aplicação de dispositivos gráficos para propósitos gerais, dentre outros fatores, causaram o aumento significativo na demanda por profissionais qualificados em computação de alto desempenho e no uso eficiente do paradigma de programação paralela. Esta demanda, combinada com a complexidade do conteúdo das disciplinas que fundamentam a computação de alto desempenho, além da dificuldade dos estudantes em compreender e visualizar o comportamento particular dos programas paralelos geram necessidade de novas práticas e recursos aplicados ao ensino do conteúdo em questão.

Outras dificuldades identificadas no ensino-aprendizagem de programação paralela derivam-se da necessidade de se consolidar conceitos de diferentes disciplinas, tais como programação, organização e arquitetura de computadores, sistemas operacionais, redes de computadores e avaliação de desempenho, fato que torna o conteúdo específico de programação paralela denso e extremamente dependente de conhecimento associado (PRASAD *et al.*, 2014).

Shamsi, Durrani e Kafi (2015) descrevem as estratégias mais recentes para o ensino de computação de alto desempenho. Neste artigo, aponta-se que, embora, existam várias práticas pedagógicas sendo aplicadas no contexto em questão, ainda não existe um modelo específico capaz de solucionar todas as dificuldades encontradas pelos alunos e professores e promover a eficiência do processo ensino-aprendizagem.

A importância do desenvolvimento de habilidades, não somente competências, para implementação de *software* paralelo robusto e de alto desempenho justificam a necessidade de estimular as capacidades individuais e coletivas dos estudantes por meio de atividades práticas de programação, sendo a proposta vigente no atual trabalho, facilitar a consolidação das virtudes apontadas por meio de desafios de programação.

3.4 Metodologias e Recursos Utilizados

Mellor-Crummey, Gropp e Herlihy (2010) apontam que, historicamente, o ensino de programação paralela tem sido fundamento nas abordagens pedagógicas tradicionais. Devido à eficácia questionável do método tradicional, novas propostas estão sendo desenvolvidas nos últimos anos, no intuito de favorecer à aprendizagem dos conceitos e motivar os estudantes.

Por meio do mapeamento sistemático realizado, somado à publicação de Bachiega *et al.* (2017), foi possível identificar as principais tendências, no que se refere às estratégias pedagógicas praticadas no contexto de programação paralela.

Percebeu-se, por meio do referido levantamento bibliográfico realizado, que o método educacional alternativo mais utilizado em programação paralela é o aprendizado ativo. Algumas das principais técnicas de ensino-aprendizagem aplicadas em programação paralela são listadas a seguir. Estes resultados podem ser verificados com melhor detalhamento na Seção 4.2.8.

- Programação por padrões;
- Feedback;
- Sala de aula invertida;
- Aprendizado baseado em projetos;
- Aprendizado baseado em problemas;
- Gameficação;
- Aprendizado baseado em modelo.

Os recursos educacionais mais utilizados em programação paralela são apresentados em sequência.

- Simuladores ou ferramentas gráficas;
- Cluster ou infraestrutura;
- Framework ou API;
- Maratona/desafio de programação.

Aqui é importante destacar as publicações de Almeida *et al.* (2012) e Giménez (2016), que constituem em duas aplicações de desafios de programação no ensino-aprendizagem de programação paralela. Ambos artigos são melhor descritos na Seção 4.3, na qual apresentam-se suas contribuições e diferenças em relação ao presente trabalho.

3.5 Considerações Finais

Neste Capítulo, foram vistos os conceitos essenciais a serem ministrados em disciplinas de programação paralela, dificuldades e lacunas no processo de ensino-aprendizagem da disciplina em questão e as principais tendências em metodologias e recursos educacionais sendo utilizados no contexto-alvo.

Inicialmente, detalhou-se o modelo curricular descrito em ACM/IEEE-CS Joint Task Force on Computing Curricula (2013), de forma a apresentar o conteúdo essencial a ser ministrado na disciplina de programação paralela e os resultados esperados derivados do aprendizado dos tópicos de ensino que compõem a grade curricular referenciada.

Em seguida, apontaram-se lacunas identificadas no processo de ensino-aprendizado do conteúdo em questão. A complexidade dos conceitos, dificuldade de visualização do comportamento dos programas paralelos, dependência de conhecimento prévio de outras disciplinas tornam o aprendizado de programação paralela desafiador e justificam novas propostas que abordem essa problemática.

Finalmente, descreveu-se, conforme o mapeamento sistemático realizado, as principais técnicas de ensino e recurso educacionais aplicadas no contexto de programação paralela. Percebe-se que as técnicas de aprendizado ativo têm sido bastante utilizadas, apresentando resultados consideráveis.

CAPÍTULO

4

TRABALHOS RELACIONADOS

4.1 Considerações Iniciais

Neste capítulo, listam-se os estudos melhores relacionados com a proposta do presente trabalho, elencados após a realização de um mapeamento sistemático e combinados com aqueles disponíveis em outro mapeamento disponível na literatura, realizado paralelamente àquele descrito neste documento.

Em um primeiro momento descrevem-se os procedimentos realizados para o desenvolvimento do mapeamento sistemático, incluindo as etapas definidas pela metodologia utilizada, segundo Kitchenham e Charters (2007). Conclui-se esta seção ao se apresentar os resultados e limitações do referido mapeamento.

Em seguida, apresentam-se os resultados finais da revisão realizada somados às informações mais relevantes provenientes do trabalho de Bachiega *et al.* (2017). Neste momento, lista-se, pontualmente, os trabalhos melhor relacionados com trabalho atual, suas contribuições e limitações.

4.2 Mapeamento Sistemático

Mapeamento sistemático pode ser definido, segundo Kitchenham (2004), como uma metodologia de pesquisa que oferece, de maneira eficiente, rígida e reproduzível, mecanismos para obtenção e classificação de material acadêmico relevante, em um determinado tópico de interesse.

O procedimento denominado revisão ou mapeamento sistemático apresenta, quando comparado com revisões *ad hoc* da literatura, resultados superiores, no que se refere a quantidade e qualidade de textos relevantes obtidos por meio do processo em questão (BRERETON *et al.*, 2007).

4.2.1 Objetivo da Revisão

O objetivo geral desta revisão sistemática trata-se da identificação do estado da arte, no que se refere à metodologias e recursos educacionais, empregados no ensino ou aprendizado das disciplinas que fundamentam a computação de alto desempenho. Desta forma, deseja-se listar os mais recentes e melhor sucedidos estudos que permitam implementar melhorias na qualidade do processo ensino-aprendizagem das disciplinas em questão.

4.2.2 Questões de Pesquisa

As questões de pesquisa, derivadas a partir do objetivo estabelecido anteriormente, são formuladas em função da estratégia PICO, que é um método capaz de guiar a definição de questões de pesquisa, a partir de quatro elementos-chave, descritos a seguir (PREGUNTA, 2007):

- ***P**opulation* (População): refere-se ao contexto ou alvo da revisão (espaço de busca);
- ***I**ntervention* (Intervenção): os elementos pontuais a serem investigados dentro do contexto (o que buscar);
- ***C**omparison* (Comparação): se cabível, define as hipóteses que se deseja comparar por meio da revisão (o que comparar);
- ***O**utcomes* (Resultados): resultados esperados, desejados ou indesejados a serem obtidos por meio da revisão.

No contexto do presente trabalho, definem-se os valores para cada um dos atributos do método como apresentado a seguir:

- ***P**opulation* (População): Disciplinas de Programação Concorrente e Programação Paralela no contexto de cursos de computação do ensino superior;
- ***I**ntervention* (Intervenção): Metodologias de ensino ou aprendizado e recursos educacionais utilizados em programação concorrente e programação paralela;
- ***C**omparison* (Comparação): não há elementos a serem comparados;
- ***O**utcomes* (Resultados): Impacto dos métodos e recursos aplicados no ensino ou aprendizado de programação concorrente e programação paralela.

As seguintes questões de pesquisa foram formuladas para este mapeamento sistemático para alcançar os objetivos definidos:

QP1: Quais métodos têm sido utilizados no ensino ou aprendizado de programação concorrente ou paralela?

QP2: Quais são os recursos educacionais utilizados para apoio ao ensino ou aprendizado das disciplinas de programação concorrente ou paralela?

QP3: Qual o impacto causado pela aplicação das referidas metodologias ou recursos educacionais utilizados?

4.2.3 Estratégias de Busca

Esta seção descreve as estratégias utilizadas para realizar as buscas de estudos primários ou secundários, incluindo as bases de pesquisa utilizadas, palavras-chave e *string* de busca definidas.

Bases de pesquisa

Utilizaram-se as bases de dados *Scopus, IEEE Xplore* e *ACM Digital Library*, por tratarem-se das fontes bibliográficas mais relevantes no contexto da computação (BRERETON *et al.*, 2007).

Palavras-chave

De forma a englobar a temática de pesquisa e os tópicos de interesse, estabeleceram-se as seguintes palavras-chave, e seus sinônimos cabíveis:

Quadro 3 – Relação de palavras-chave e sinônimos

Concurrent Programming	*Concurrent Computing*
Parallel Programming	*Parallel Computing*
Teach	*Educational Resource*, *Learn*, *ER*

String de busca

A *string* de busca utilizada para pesquisa nas bases de dados consideradas, definida a partir da concatenação das palavras-chave e seus respectivos sinônimos, é apresentada a seguir:

Quadro 4 – *String* de busca

(“concurrent programming” OR “concurrent computing” OR “parallel programming” OR “parallel computing”) AND (“teach*” OR “learn*” OR “educational resource” OR “ER”)

A *string* de busca, apresentada no Quadro 4, foi submetida nos mecanismos de pesquisa das bases de dados definidas anteriormente, obtendo-se um conjunto de estudos, classificado pela quantidade de material retornado por cada uma das bases. Tais valores são representados por meio da Tabela 1.

Tabela 1 – Resultado da pesquisa.

Base de Dados	**Resultado**
Scopus	370
IEEE Xplore	127
ACM Digital Library	198
Proquest	2
Total	697
Total de estudos sem repetição	515

Utilizou-se a ferramenta *Mendeley* (<https://www.mendeley.com/>) para obtenção da quantidade de estudos sem repetição, por meio da exportação do conjunto de dados extraído de cada uma das bases, em formato .bib, para a referida ferramenta, que possui um mecanismo interno para tratamento de duplicatas.

4.2.4 Seleção

Nesta seção definem-se os critérios que determinam a inclusão ou exclusão dos textos obtidos das bases de pesquisa. Desta forma, o Quadro 5 ilustra os parâmetros estabelecidos para a consideração dos estudos analisados.

Quadro 5 – Relação dos critérios de inclusão e exclusão

Critérios de Inclusão	**Critérios de Exclusão**
Estudos relacionados ao ensino/aprendizado de programação concorrente ou paralela.	Estudos que não tenham relação com os temas de ensino de programação concorrente ou paralela.
Estudos relacionados a recursos educacionais aplicados ao ensino/aprendizado de programação concorrente ou paralela.	Estudos incompletos (chamadas de evento, resumos, etc).
-	Estudos que não sejam focados em quaisquer subáreas da computação.
-	Estudos classificados como literatura cinza.
-	Estudos cujo idioma não seja inglês.
-	Recursos propostos no estudo não estejam disponíveis publicamente.

4.2.5 Avaliação

Utilizou-se uma metodologia para refinamento e classificação dos textos encontrados em cada uma das fontes pesquisadas, conforme proposto por Kitchenham e Charters (2007). O processo é descrito, em detalhes, a seguir.

Metodologia de Avaliação

O procedimento de refinamento ocorre em três fases, nas quais aplicam-se, sucessivamente, os critérios de inclusão e exclusão definidos anteriormente, em trechos distintos dos documentos retornados pela busca inicial. Desta forma, por meio da execução de cada uma das etapas do processo, é possível eliminar os estudos não condizentes com o objetivo da revisão.

- **Fase I: Leitura dos títulos e resumos**: Analisam-se, primeiramente, o título e resumo de cada um dos textos retornados. As análises consistem na aplicação dos critérios de inclusão e exclusão definidos, de forma em que seja possível obter somente estudos relacionados ao objeto de pesquisa;
- **Fase II: Leitura dos resultados e conclusões**: A partir dos resultados da Fase I, aplicam-se novamente os critérios de inclusão e exclusão após a leitura dos resultados e conclusões de cada documento, com o mesmo objetivo estabelecido na etapa anterior;
- **Fase III: Leitura dos estudos completos**: Para concluir o processo, os estudos remanescentes são analisados completamente, considerando todo seu conteúdo. Após a realização desta etapa, obtém-se a relação final dos textos considerados na revisão.

A Tabela 2 ilustra a seleção de estudos, por critérios de inclusão e exclusão, em cada uma das fases realizadas.

Tabela 2 – Estudos selecionados após a aplicação dos critérios de inclusão e exclusão.

Fase	Quantidade de Estudos Selecionados
I	256
II	136
III	93

4.2.6 Avaliação de Qualidade

Dybå e Dingsøyr (2008) sugerem uma relação de atributos binários para avaliação da qualidade dos estudos analisados, considerando a relevância destes textos, em um mapeamento ou revisão sistemática. Com base nesta fundamentação estabeleceram-se os critérios de qualidade, representados por meio do Quadro 6.

Quadro 6 – Critérios de qualidade

Número	Critério
1	O estudo é baseado em pesquisa (ou trata-se de um relatório de lições aprendidas)?
2	Os objetivos são bem definidos?
3	Foi desenvolvida ou utilizada uma metodologia de ensino diferente do modelo tradicional?
4	Foram desenvolvidos novos recursos de ensino?
5	Os resultados são exibidos de forma clara?
6	O trabalho é conclusivo?
7	O estudo é reprodutível?
8	O trabalho tem comprovação estatística?

Atribui-se, para cada um dos parâmetros estabelecidos e em cada um dos estudos avaliados, o valor 1 (sim) ou 0 (não). Por fim, os valores para cada estudo são somados e os trabalhos são, então, classificados, em função de sua qualidade.

A classificação final pode ser vista em <http://bit.do/pontuacao_MS>.

4.2.7 Extração de Dados

Para conduzir o processo de extração de dados, foi desenvolvido um formulário que contém a relação dos atributos considerados mais relevantes, retirados de cada um dos estudos considerados.

O endereço <http://bit.do/dadosbrutos_MS> possui a compilação de todas as informações extraídas, que não serão apresentadas, em extensão, no presente trabalho por restrições de espaço.

O formulário de extração de dados utilizado pode ser visto por meio do Quadro 7.

Quadro 7 – Formulário de Extração de dados

Atributo	**Descrição**
Índice	Índice numérico para referência
Data	Data de extração dos dados
Autor	Nomes dos autores
Título	Título da publicação
Ano	Ano da publicação
Abstract	Resumo extraído do próprio estudo
Keywords	Palavras-chave extraídas do próprio estudo
Fonte	Onde foi publicado o estudo
Base	Base de dados onde obteve-se a publicação
Tipo	Tipo da publicação (artigo, livro, *paper* de conferência, *paper* de *workshop*)
Abordagem	Qual a metodologia e recursos utilizados
Tecnologia	Quais são as tecnologias (*frameworks*, APIs, linguagens) utilizadas
Conclusões	Quais as conclusões e impacto da publicação

4.2.8 Resultados

Conforme discutido na Seção 4.2.5, 93 trabalhos foram selecionado após a aplicação sucessiva dos critérios de inclusão e exclusão. Tais estudos foram, posteriormente, classificados conforme sua relevância, baseados nos critérios definidos na Seção 4.2.6. As informações extraídas deste conjunto final de documentos são apresentadas na presente seção.

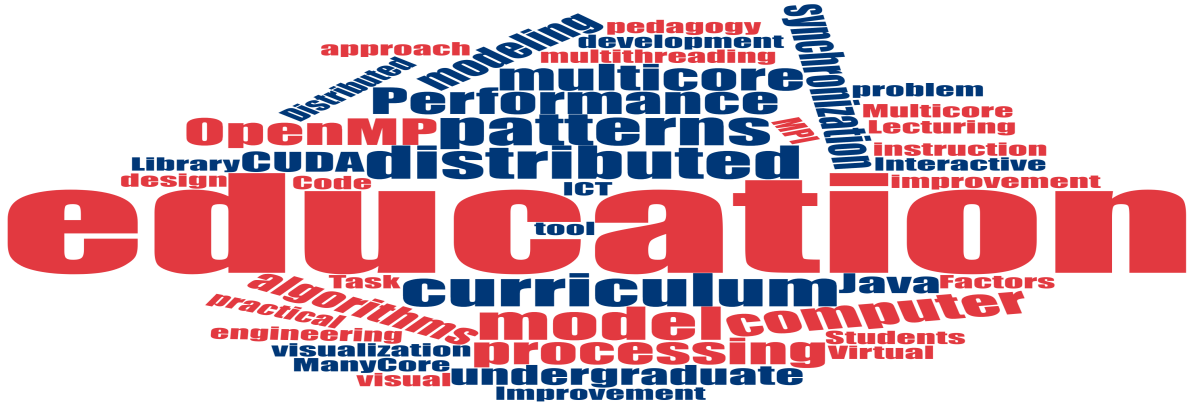
A Figura 12 apresenta a relação do número de publicações por ano.

Figura 12 – Quantidade de estudos publicados por ano

Percebe-se com a Figura 12 o surgimento do interesse pela temática do ensino-aprendizado das disciplinas de programação concorrente ou paralela no início da década de 1990, e seu aumento, considerável, no início da década de 2010, fato primariamente atribuído ao desenvolvimento de novas tecnologias, tais como CUDA e as arquiteturas *many-core*; e ao aumento da importância da HPC para construção de soluções para problemas complexos e críticos.

No que se refere às palavras-chave extraídas, propomos um gráfico do tipo *word cloud* para ilustrar os atributos de indexação mais comuns. A Figura 13 representa a frequência em que cada uma das palavras-chave foi utilizada para caracterizar as publicações analisadas. Assim, o tamanho relativo de cada uma das expressões ilustradas representa o quão frequente ela foi utilizada.

Figura 13 – Relação das *keywords* mais frequentes



É possível notar, por meio da análise do gráfico apresentado, que as palavras-chave mais comuns são aquelas que definem a abstração abordada (*parallel computing/programming*, seguidas daquelas relacionas à educação (*education, teaching*) e, por fim, das expressões que definem tecnologias, tais como CUDA ou OpenMP. Este recurso ilustra, parcialmente, o foco das publicações analisadas. Percebeu-se que a terminologia *parallel programming/computing* é a mais utilizada para descrever as disciplinas e abstrações de mesmo nome e também àquelas classificadas como programação ou computação concorrente.

Também contabilizamos a quantidade de publicações indexadas por base de dados. A Tabela 3 ilustra o que foi obtido.

Tabela 3 – Quantidade de Publicações por base de dados

Base	**Quantidade de Publicações**
IEEE Xplore	35
ACM Digital Library	33
Scopus	23
Proquest	2

Classificou-se a quantidade de publicações por tipo do veículo de divulgação: periódico, conferência/simpósio, *workshop* e capítulo de livro (vide Tabela 4).

A Tabela 4 representa graficamente aquilo que foi obtido.

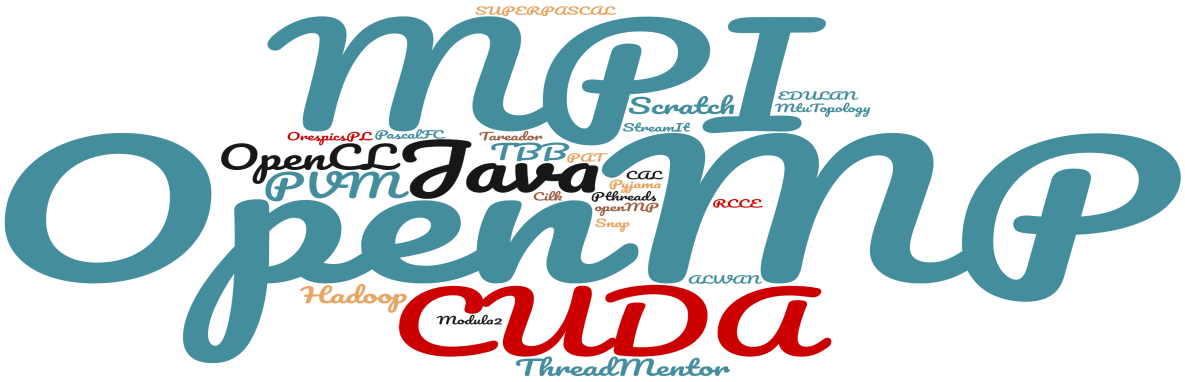
Tabela 4 – Quantidade de Publicações por tipo

Base	**Quantidade de Publicações**
Conferência	70
Artigo	21
Capítulo de Livro	2

Percebe-se, por meio da interpretação das informações contidas nas tabelas anteriormente apresentadas, que o meio mais comum de divulgação dos estudos considerados por este trabalho trata-se, primeiramente, de conferências e, em sequência, periódicos. Também é possível perceber que os eventos coordenados e periódicos publicados por intermédio da IEE e ACM são os ambiente mais frequentes de divulgação dos trabalhos aqui discutidos.

Julgou-se procedente relacionar a utilização de tecnologias no contexto do ensino-aprendizado, desta forma estabelecendo as principais linguagens de programação, *frameworks*, APIs e ferramentas, em geral, empregadas no contexto nas disciplinas-alvo. As referidas tecnologias tratam-se tanto dos objetos de ensino das disciplinas em questão, quanto os recursos de apoio à estratégia pedagógica praticada. A Figura 14 representa o agrupamento, por palavras-chave, elaborado.

Figura 14 – Relação das tecnologias mais utilizadas no âmbito do ensino de programação paralela



Percebe-se na Figura 14 a maior utilização da interface de programação OpenMP em grande parte dos trabalhos indexados, seguida pelo MPI e CUDA. Em critérios de linguagem de programação, verificou-se a maior utilização de Java.

Finalmente, outro aspecto que julgou-se importante classificar, trata-se da abordagem praticada no trabalho, ou o que foi desenvolvido e apresentado pela publicação. Para sintetizar o conteúdo, os estudos foram agrupados em duas categorias: aqueles que propõem metodologias de ensino para a disciplina de programação concorrente e paralela e aqueles que propõem recursos educacionais, aplicáveis no contexto de ensino da mesma disciplina. Desta forma, a Tabela 5 ilustra as informações obtidas.

Tabela 5 – Quantidade de Publicações por abordagem desenvolvida

Abordagem	**Quantidade de Publicações**
Metodologia de ensino	63
Recursos educacionais	30

Nota-se que em significante parte das publicações os autores descrevem metodologias para o ensino do conteúdo foco deste trabalho. São, comparativamente, reduzidas as ferramentas de apoio ao ensino, ou recursos educacionais, que foram desenvolvidos neste âmbito, especialmente aqueles que estão publicamente disponíveis, o que pode apontar para a necessidade de esforço e pesquisa neste tema específico.

4.2.9 Limitações

Segundo Kitchenham e Charters (2007), o estabelecimento das questões de pesquisa e *string* de busca, embora fundamento em uma metodologia criteriosa e concisa, pode, eventualmente, gerar incoerências, polarização (*bias*) e pode-se excluir, de forma não proposital, resultados relevantes do material resultante do mapeamento ou revisão sistemática.

Desta forma, assume-se a possibilidade de erros derivados da subjetividade inerente aos processos que constituem o método, admitindo, por fim, a eventualidade de ignorância de estudos que podem, porventura, serem válidos para o contexto abordado.

A avaliação por critérios de qualidade, conforme estabelecido, foi baseada no trabalho de Dybå e Dingsøyr (2008), escolhido pela simplicidade do método descrito e pela relativa qualidade dos resultados obtidos. O procedimento em questão não constitui em uma forma única e inquestionável de classificação, e sua própria natureza binária constitui em um limitante para a validade e completude das informações obtidas.

4.3 Conclusões

Por intermédio da realização do mapeamento sistemático descrito nas seções anteriores deste capítulo, identificou-se um conjunto de trabalhos que ilustra o estado da tarde, no que se refere às metodologias e recursos educacionais empregados no ensino de programação paralela e disciplinas equivalentes.

É razoável afirmar que o objetivo da revisão, conforme definido na Seção 4.2.1, fora satisfeito, havendo, portanto, soluções para cada uma das questões de pesquisa propostas em 4.2.2.

Verificou-se, por meio da análise da relação final dos estudos obtida, a relação das práticas de ensino e aprendizado empregadas no contexto da disciplina de programação paralela. As principais estratégias empregadas são ilustradas por meio da Tabela 6.

Tabela 6 – Quantidade de publicações por método

Método	**Quantidade**
Aprendizado baseado em problemas	**9**
Programação por padrões	8
Aprendizado orientado a exemplo	5
Aprendizado baseado em modelo	4
Sala de aula invertida ou parcialmente invertida	3
Aprendizado baseado em projetos	**2**
Gameficação	2
Aprendizado baseado em módulos	2
Aprendizado baseado em pesquisa	2

A Tabela 6 apresenta a quantidade de estudos obtidos, indexados por estratégia (técnicas, métodos, abordagens) educacional utilizada. As tuplas em negrito referem-se aos tópicos de interesse no contexto do primeiro trabalho. Destaca-se que os demais resultados, não contabilizados na tabela, referem-se às metodologias tradicionais de ensino, que não são o foco do presente trabalho, ou não descrevem uma estratégia de ensino/aprendizado específica.

As linhas destacadas na 6 ilustram a importância das temáticas abordadas neste trabalho e a relevância dos estudos associados obtidos. É visto, portanto, que as técnicas associadas ao aprendizado ativo de programação paralela representam a principal estratégia educacional não tradicional aplicada no contexto de programação paralela. As publicações melhor relacionadas com esta monografia serão descritas na Seção 4.3.1.

Adicionalmente, apresenta-se, nos Quadros 8, 9, 10 e 11, a classificação das publicações por abordagem. Os referidos quadros foram fracionados para favorecer a leitura das informações contidas nos mesmos.

Quadro 8 – Classificação das publicações por abordagem. Parte I

Tradicional	Baseada em Problema	Baseada em Projeto	Orientada a modelo
(MURESANO; REXACHS; LUQUE, 2012)	(IPARRAGUIRRE; FRIEDRICH; COPPO, 2012)	(LUPO; WOOD; VICTORINO, 2012)	(LIN; TATAR, 2011)
(GROSS, 2011)	(STRAZDINS, 2012)	(BURKHART; GUERRERA; MAFFIA, 2014)	(CARRO; HERRANZ; MARINO, 2013)
(RAGUE, 2011)	(GIMÉNEZ, 2016)		(SAPARKHOJAYEV, 2013)
(ACACIO *et al.*, 2012)	(CUENCA; GIMéNEZ, 2016)		(DOLGOPOLOVAS *et al.*, 2015)
(BROWN; LU; MIDKIFF, 2013)	(WHITE *et al.*, 2012)		
(TSIOPOULOS *et al.*, 2014)	(BERNABÉ *et al.*, 2014)		
(LIU, 2017)	(YAZICI; MISHRA; KARAKAYA, 2016)		
	(ZAKHAROVA; ZAKHAROV, 2018)		
	(CHEN *et al.*, 2018)		

Quadro 9 – Classificação das publicações por abordagem. Parte II

Colaborativa	Baseada em Pesquisa	Programação por Padrões	Gamificação
(MANOGARAN, 2013)	(De Freitas, 2013)	(ARROYO, 2013)	(KITCHEN; SCHALLER; TYMANN, 1992)
	(GIACAMAN; SINNEN, 2014)	(ADAMS, 2014)	(FRESNO *et al.*, 2016)
		(BROWN *et al.*, 2014)	
		(CAPEL; TOMEU; SALGUERO, 2017b)	
		(ADAMS; BROWN; SHOOP, 2013)	
		(FERNER; WILKINSON; HEATH, 2013)	
		(FERNER; WILKINSON; HEATH, 2014)	
		(CAPEL; TOMEU; SALGUERO, 2017a)	

Quadro 10 – Classificação das publicações por abordagem. Parte III

Orientado a exemplo	Baseado em módulos	Sala de aula invertida	Sala de aula parcialmente invertida
(LIU *et al.*, 2011)	(BROWN; SHOOP, 2012)	(ZARESTKY; BANGERTH, 2014)	(GROSSMAN *et al.*, 2017)
(VALENTINE, 2014)	(BURTSCHER *et al.*, 2015)	(MOORE; DUNLOP, 2016)	
(FELDHAUSEN; BELL; ANDRESEN, 2014)			
(SHAMSI; DURRANI; KAFI, 2015)			
(TOMEU-HARDASMAL; SALGUERO; CAPEL, 2015)			

Quadro 11 – Classificação das publicações por abordagem. Parte IV

Abordagem analítica	Feedback	Sem classificação	Nova abordagem
(UL-ABDIN; SVENSSON, 2015)	(JOINER *et al.*, 2006)	(LI; GUO; ZHENG, 2008)	(SKOPIN, 2014)
		(TORBERT *et al.*, 2010)	(AYGUADÉ *et al.*, 2015)
		(KO; BURGSTALLER; SCHOLZ, 2013)	(EIJKHOUT, 2016)
		(RIVOIRE, 2010)	(PRAUN, 2011)
		(FENG; TILEVICH; FENG, 2015)	(SADOWSKI *et al.*, 2011)
			(LEVANDOWSKI; PEROULI; BRYLOW, 2019)
			(RADENSKI, 2012)

Posteriormente, determinaram-se os principais recursos educacionais empregados no âmbito de programação concorrente. As principais estratégias empregadas são ilustradas por meio da Tabela 6.

Tabela 7 – Quantidade de publicações por recurso educacional

Método	**Quantidade**
Simuladores ou ferramentas gráficas	14
Cluster ou infraestrutura	5
Framework ou API	3
Repositório ou curso online	4
Correção automática	2
Maratona/desafio de programação	**2**

Apresenta-se a classificação das publicações referentes à recursos educacionais por tipo nos Quadros 12 e 13, também fracionados para facilitar a leitura do conteúdo dos mesmos.

Quadro 12 – Classificação dos Recursos Educacionais por Tipo : parte I

Simuladores ou ferramentas gráficas	Cluster ou infraestrutura	Framework ou API
(BOSSCHERE, 1989)	(FOLEY; HURSEY, 2015)	(IMAM; SARKAR, 2014)
(BURKHART, 1997)	(CAPRETTI *et al.*, 2001)	(DAVIES, 1990)
(RAGUE, 2012)	(RADENSKI, 2012)	(DANNER; NEWHALL; WEBB, 2019)
(CARR *et al.*, 2002)	(BUI *et al.*, 2013)	
(AYGUADÉ *et al.*, 2015)	(AZIZ *et al.*, 2015)	
(BI; BEIDLER, 2007)		
(ABDULSALAM, 2009)		
(CARR; MAYO; SHENE, 2003)		
(SADOWSKI *et al.*, 2011)		
(VALLS-VARGAS; ZHU; ONTAÑÓN, 2017)		
(CARR *et al.*, 2003)		
(POPOVIĆ; VLADIMIR; ŠILIĆ, 2018)		
(ADAMS *et al.*, 2018)		

Quadro 13 – Classificação dos Recursos Educacionais por Tipo : parte II

Repositório/curso online	Correção automática	Maratona/desafio de programação
(FOLEY *et al.*, 2017)	(AZIZ *et al.*, 2015)	(GIMÉNEZ, 2016)
(ADAMS, 2015)	(GROSSMAN *et al.*, 2017)	(ALMEIDA *et al.*, 2012)
(SARKAR *et al.*, 2017)		
(CHAUDHURY *et al.*, 2018)		

No que se refere aos impactos causados pela aplicação das estratégias e ferramentas descritas anteriormente, verificam-se resultados positivos e relevantes, tais como o aumento do interesse pela disciplina em questão, redução da taxa de evasão nas aulas relacionadas, aumento do desempenho dos alunos em qualidade de *software* desenvolvido e aumento da proporção de aprovação dos estudantes. É importante destacar que, por se tratarem de resultados fundamentados na teoria pedagógica e suas aplicações, existe, naturalmente, subjetividade na interpretação do produto das pesquisas desenvolvidas que, essencialmente, é validado por meio de testes, questionários, avaliações e outros mecanismos sujeitos às opiniões, virtudes e vícios humanos.

4.3.1 Principais Trabalhos

Nesta seção, descrevem-se os trabalhos com interseção mais significativa em relação à presente dissertação, em ordem cronológica e de relevância.

O trabalho de Leal e Silva (2003) descreve o sistema de apoio à realização de maratonas de programação *Mooshak*, utilizado como base para realização da competição espanhola de programação paralela. Neste estudo, as particularidades e funcionamento da ferramenta são descritas em extensão, assim como suas vantagens e diferenças em relação à *software* similares. As instruções referentes às funcionalidades da ferramenta, apresentadas neste artigo, foram amplamente utilizadas como referência para o uso do sistema em questão.

O artigo de Almeida *et al.* (2012) detalha a utilização dos desafios de programação do concurso de programação espanhol como recurso educacional, empregado como apoio ao ensino da disciplina de programação paralela. Neste trabalho apresenta-se o processo de adaptação da ferramenta *Mooshak* para o uso no contexto educacional, a maneira como foram utilizados os problemas da referida maratona de programação em sala de aula, objetos de ensino associados desenvolvidos e os impactos da aplicação dos recursos em questão nas disciplinas de duas universidades espanholas: Murcia e La Laguna. Trata-se do primeiro relato da utilização de desafios ou maratonas de programação para o ensino-aprendizado de programação paralela. Embora constitua em um *short paper* e um trabalho em andamento (no momento de sua publicação), os resultados parciais obtidos e a documentação desenvolvida, disponível publicamente, tratam-se de informações essenciais para o desenvolvimento do presente trabalho.

Na publicação de Lupo, Wood e Victorino (2012), apresenta-se uma metodologia para o ensino de programação paralela, considerando conteúdos das disciplinas de programação paralela e computação gráfica. A abordagem apresentada contém um modelo orientado a projetos, no qual une-se conceitos específicos das disciplinas mencionadas, de forma a estimular os estudantes a projetarem sistemas gráficos e de alto desempenho. Nota-se que a mescla dos conceitos de ambas disciplinas favorece o aprendizado prático e interesse dos alunos, quando submetidos a aplicações reais do conteúdo. O método apresentado neste trabalho, devido a seus resultados favoráveis e de sua descrição detalhada, possibilita a replicação de certos elementos no contexto do trabalho atual, no intuito de introduzir aspectos dinâmicos e motivadores ao aprendizado da disciplina em questão.

O artigo de De Freitas (2013) relata o desenvolvimento de uma metodologia para ensino de programação paralela em dispositivos híbridos, baseada em projetos de pesquisa, financiados por agências de pesquisa nacionais. Demonstra como são abordados tópicos como arquiteturas híbridas, *benchmarking* e balanceamento de carga. Os resultados indicam um bom desempenho dos participantes, aliado a altos valores de motivação e interesse pelo curso e conteúdo ministrado.

O trabalho de Fresno *et al.* (2016) descreve uma metologia inédita para ensino de programação paralela em multiprocessadores heterogêneos utilizando a estratégia de gameficação.

Propõe-se desafios que fundamentam o conteúdo da disciplina, que são pontuados e avaliados em uma estratégia similar às competições de jogos eletrônicos. O autor apresenta as informações relativas à satisfação dos estudantes em função da metodologia proposta. O uso de desafios como critério de avaliação aplicado em ambiente competitivo apresentado neste trabalho é particularmente relevante para o desenvolvimento de uma estratégia fundamentada na utilização de desafios e maratonas de programação como recurso de apoio ao aprendizado da disciplina-alvo deste trabalho.

O artigo de Giménez (2016) detalha um método de ensino para a disciplina de programação paralela, essencialmente prático, baseado nos problemas da maratona de programação paralela da Espanha (*spanish parallel programming contest*). Descreve-se a abordagem, com plano de aulas e recursos didáticos associados para integrar conceitos de programação paralela em dispositivos *multi-core* e híbridos. Os resultados obtidos com a aplicação da metodologia descrita pelo artigo na universidade de Murcia, Espanha, incluem o aumento da taxa de satisfação e aprovação dos estudantes na disciplina em questão. Esta publicação descreve uma proposta bastante similar àquela definida no presente trabalho, sendo este o estudo melhor relacionado com a concepção vigente nesta dissertação. Portanto, o processo de construção e aplicação do método descrito no artigo referenciado constitui em um aspecto balizador do nosso trabalho.

Cabe-se destacar algumas diferenças em relação ao trabalho descrito anteriormente e o presente projeto, no intuito de explicitar algumas contribuições do último.

Na publicação referenciada, há a construção de um curso completo de programação concorrente, baseado em projetos, no qual é destacado o processo de elaboração, plano de aula e ementa do mesmo. As particularidades e impacto do uso dos desafios de programação, assim como a metodologia de aprendizado que fundamenta o curso, não constituem em um foco do referido trabalho.

4.4 Considerações Finais

Neste capítulo descreveu-se, extensivamente, o processo de seleção dos trabalhos melhores relacionados à proposta vigente no presente trabalho, fundamentado na metodologia de revisão ou mapeamento sistemático da literatura.

Mostrou-se que pesquisas no contexto de ensino e aprendizado de programação paralela existem desde a década de 1980 e o interesse pelo tema aumentou consideravelmente recentemente, especialmente depois da década de 2010, devido, principalmente, ao desenvolvimento de novas tecnologias, como a plataforma *CUDA*, e o surgimento de arquiteturas *many core*.

O aprendizado ativo e técnicas educacionais alternativas constituem na maior porção das publicações obtidas por meio do mapeamento sistemático. O aumento do desempenho e motivação dos alunos tratam-se dos principais resultados relevantes divulgados através destes

estudos, o que justifica o interesse e novas pesquisas neste contexto.

Apresentou-se, por fim, os trabalhos melhor relacionados ao tema de pesquisa e à proposta vigente na presente monografia. Destacam-se os artigos de Almeida *et al.* (2012) e Giménez (2016), que descrevem aplicações dos desafios de programação como recursos de apoio ao ensino-aprendizado de programação paralela. As estratégias detalhadas neste conjunto de estudos fundamentam e justificam o planejamento proposto e contribuições esperadas do presente trabalho.

CAPÍTULO

5

RESULTADOS

5.1 Considerações Iniciais

Neste capítulo, os resultados derivados da execução do presente trabalho são apresentados.

Em um primeiro momento, traz-se uma visão geral sobre os experimentos realizados.

Nas Seções subsequentes, os experimentos são detalhados e os resultados provenientes de sua realização, assim como demais artefatos, são exibidos e analisados.

Finalmente, comparam-se os resultados entre os experimentos e faz-se uma análise global no intuito de confrontar o que foi obtido em função daquilo que foi proposto inicialmente, almejando, consequentemente, validar as hipóteses e elencar as contribuições do presente trabalho.

5.2 Relação de Experimentos

No intuito de avaliar as hipóteses estabelecidas e satisfazer os objetivos propostos neste trabalho, foram realizados um total de 4 experimentos, os quais foram caracterizados de formas distintas em função de seu planejamento, configuração, publico alvo e objetivos específicos.

- Experimento I: Disciplinas de Programação Concorrente para Engenharia da computação e Bacharelado em Ciência da Computação baseadas em *Team Based Learning* (2018);
- Experimento II: Curso de Difusão Cultural de Introdução ao *OpenMP* para alunos iniciantes em computação (2019);
- Experimento III: Curso de Difusão Cultural de Introdução ao *OpenMP* baseado em *Problem Based Learning* (2019);

- Experimento IV: Disciplina de Computação de Alto Desempenho para Engenharia de Computação baseada em *Problem Based Learning* (2019).

Todos os experimentos indicados foram conduzidos em sala de aula, presencialmente, com diferentes caracterizações dos participantes (amostras), conteúdo didático, recursos educacionais e metodologias de ensino/aprendizado empregadas. O intuito da variabilidade dos fatores dos experimentos foi estabelecer a relação e impacto de diferentes metodologias e recursos educacionais sobre turmas de composição heterogênea. Além disso, busca-se analisar a contribuição do uso de desafios de programação sobre a qualidade do processo pedagógico e sobre o engajamento dos alunos.

Foram desenvolvidos um total de 58 desafios de programações paralela, que foram utilizados nos experimentos desta dissertação. Estes desafios, contendo código fonte, descrição e casos de teste, foram implementados a partir do material disponibilizado pelo professor orientador e pelo autor do presente trabalho. Adicionalmente, outros 9 desafios, que não foram utilizados no contexto deste trabalho, foram implementados pelo aluno Gabriel Martins, em razão de seu trabalho de conclusão de curso. Além disso, o referido aluno da graduação também contribuiu para a padronização dos códigos-fonte existentes, suas descrições e casos de teste. O conjunto final com 67 desafios pode ser obtido no *link* <http://tiny.cc/desafios_programacao_para>.

Foram produzidos, no contexto dos experimentos III e IV, material didático composto de descrições e códigos-fonte sequenciais para desafios de programação, além de síntese teórica necessária para resolução dos referidos desafios. Este material pode ser obtido no endereço <http://tiny.cc/cadernos_de_desafios>.

Com exceção do experimento I, foram aplicados pré-testes e pós-testes para mensurar a aborção do conhecimento teórico pelos estudantes. Todas as avaliações encontram-se disponíveis em <http://tiny.cc/pre_e_pos_testes>.

De uma forma geral, o desempenho dos participantes dos referidos experimentos foi estabelecido em função da **avaliação quantitativa do conhecimento teórico, análise de corretude e qualidade dos códigos-fonte desenvolvidos e avaliação qualitativa da satisfação dos alunos** em relação ao método pedagógico aplicado, conteúdo ministrado e curso, de uma forma global. Além disso, definiu-se um conjunto de critérios com intuito de compor uma pontuação ponderada para cada um dos parâmetros estabelecidos. Estes valores permitem a comparação entre os experimentos e posterior análise sobre os resultados obtidos. Os referidos critérios e respectivos pesos atribuídos são:

- **Uso correto de paralelismo**: explorar corretamente aspectos paralelos do problema/aplicação. Equivale a 40% do valor da média final ponderada.
- **Uso correto do modelo de programação**: uso adequado da plataforma disponível. Equivale a 30% do valor da média final ponderada.

- **Saída correta/esperada**: resultado correto e desempenho da solução. Equivale a 20% do valor da média final ponderada.

- **Legibilidade e boas práticas de programação**. Equivale a 10% do valor da média final ponderada.

Adicionalmente, o planejamento dos experimentos, no que se refere ao conteúdo abordado em sala de aula, foi derivado da análise sobre o ensino de programação paralela, apresentada no Capítulo 3, e, sobretudo, na estrutura curricular proposta em ACM/IEEE-CS Joint Task Force on Computing Curricula (2013) e expandida em Prasad *et al.* (2014).

O processo de planejamento, exposição e análise dos experimentos é fundamentado na publicação de Wohlin *et al.* (2012), visando permitir a reprodutibilidade dos mesmos e garantir o rigor científico na análise dos resultados obtidos. A partir da referida proposta, a seguinte organização foi definida:

- Escopo;

- Planejamento;

- Execução;

- Resultados;

- Análise.

Os métodos estatísticos empregados nesta dissertação, afim de validar cientificamente as hipóteses estabelecidas, foram selecionados conforme as recomendações descritas em Armitage, Berry e Matthews (2008).

. Este mecanismo prevê a categorização da motivação de um indivíduo em quatro elementos: Atenção (A), Relevância (R), Confiança(C) e Satisfação (S), cuja avaliação permite estabelecer conclusões acerca da receptividade do referido indivíduo a um evento de cunho pedagógico.

No contexto dessa dissertação, utiliza-se o instrumento CIS (*Course Interest Survey*), proposto por Keller (2010). O CIS trata-se de um questionário composto de 34 questões de múltipla escolha em escala *likert* (de 1 a 5 - muito ruim a muito bom).

Nas próximas Seções, os experimentos listados previamente são detalhados e seus resultados são exibidos juntamente com as análises e avaliações pertinentes, estruturados em função das atividades elencadas anteriormente.

5.3 Experimento I: Disciplinas de Programação Concorrente Utilizando Team Based Learning

Este experimento, realizado no período de agosto a dezembro de 2018, contemplou três turmas distintas da disciplina obrigatória de Programação Concorrente, do ICMC USP. Os alunos participantes cursavam Bacharelado em Ciência de Computação, Engenharia da Computação ou eram alunos do programa de pós-graduação em Ciências de Computação e Matemática Computacional. O experimento foi realizado em sala de aula e conduzido por um professor e por três alunos do Programa de Aperfeiçoamento do Ensino.

O objetivo geral deste experimento foi avaliar o aprendizado dos estudantes, quando submetidos à desafios práticos de programação em conjunto da metodologia de aprendizado colaborativo *Team Based Learning*. Os desafios utilizados neste experimento constituíram uma atividade prática, ao fim de cada aula, correspondente à etapa de aplicação do *TBL*. Verificaram-se resultados promissores, considerando os critérios estabelecidos na seção 5.2.

5.3.1 Escopo

Objetivo

O objetivo deste experimento é avaliar a capacidade dos alunos em uma disciplina obrigatória em computação, utilizando da metodologia de aprendizado colaborativo *Team Based Learning* e desafios de programação, de desenvolver códigos paralelos corretos, de qualidade e com desempenho, além de aferir a absorção do conhecimento teórico pelos estudantes participantes do experimento.

Deseja-se avaliar o processo de aprendizado e a influência da metodologia aplicada (TBL) em conjunto com o uso de desafios de programação.

Objeto

O objeto de estudo deste experimento é o uso de desafios de programação, aliados à metodologia de aprendizado colaborativo *Team Based Learning* para o ensino de programação paralela.

Foco qualitativo

O foco qualitativo do presente experimento é estabelecer o desempenho dos alunos, considerando os critérios estabelecidos na seção 5.2, por atividade e em função dos modelos de programação *OpenMP, MPI e CUDA*. Além disso, visa-se mensurar a absorção do conhecimento teórico referente ao conteúdo ministrado e a motivação dos alunos em função da qualidade das aulas oferecidas.

Perspectiva

Por se tratar de um estudo experimental de cunho pedagógico, cujo objeto é um recurso

educacional associado a uma metodologia de ensino/aprendizado, a perspectiva é que este experimento seja conduzido em um ambiente de sala de aula e, considerando a natureza prática visando ao desenvolvimento de algoritmos paralelos, é ideal que o experimento seja aplicado em um meio com infraestrutura computacional adequada.

Contexto

Em função da perspectiva definida, o contexto deste experimento foi a disciplina de Programação Concorrente, oferecida como matéria obrigatória para os cursos de Engenharia da Computação (SSC0742) e Bacharelado em Ciências de Computação (SSC0143) do ICMC USP nos 7º e 6º período da graduação, respectivamente. O estudo foi realizado no segundo semestre de 2018, com duração entre os meses de agosto e dezembro, com a participação de um total de 151 alunos, divididos entre 3 turmas e provenientes dos cursos de Engenharia da Computação, Ciência de Computação e pós-graduação em Ciências de Computação e Matemática Computacional. A composição das turmas pode ser vista na Tabela 8.

Tabela 8 – Alunos por curso: Experimento I

Curso	Quantidade de Alunos
Engenharia da Computação	34
Bacharelado em Ciência de Computação	111
Pós-Graduação	6

Além disso, o estudo foi desenvolvido em sala de aula, na qual os estudantes utilizaram seus computadores pessoais e, em algumas ocasiões, foi utilizado um laboratório de ensino com computadores pré-configurados com as ferramentas de desenvolvimento de código e demais recursos de *software* e *hardware* adequados.

5.3.2 Planejamento

Definição das Questões de Pesquisa

Em função dos objetivos estabelecidos previamente, definiram-se as seguintes questões de pesquisa:

QP1 Os alunos foram capazes de aprender o conteúdo teórico de *OpenMP*, *MPI* e *CUDA* (análise das avaliações em grupo, em etapa da metodologia)?;

QP2 Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP*, *MPI* e *CUDA* corretas e com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho?; e

QP3 Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?

Seleção de Sujeitos

Os sujeitos selecionados para o presente experimento constituíram em alunos regularmente matriculados na disciplina de programação concorrente, distribuídos conforme ilustrado na Tabela 8.

Naturalmente, os pesquisadores não exerceram influência sobre a seleção destes alunos, que foram matriculados segundo as normas da Universidade de São Paulo e, portanto, a amostra pode ser considerada composta de maneira aleatória, dentre a população de alunos de computação.

Descrição da Instrumentação

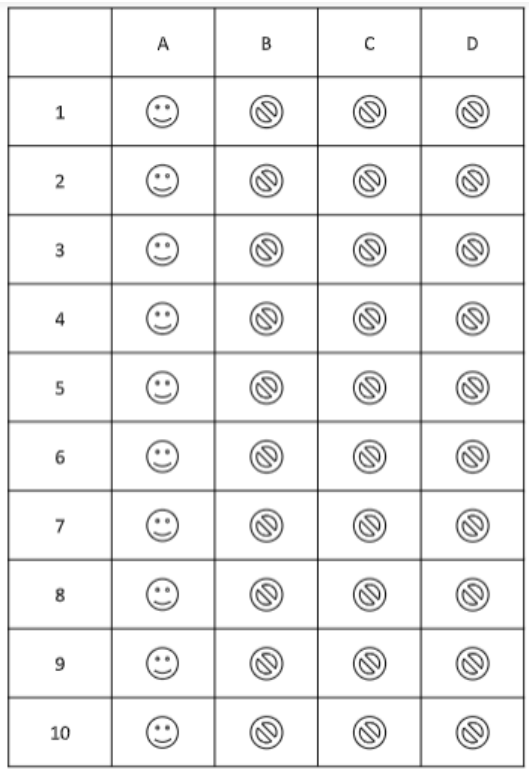
Este experimento foi planejado como uma disciplina obrigatória fundamentada na metodologia de aprendizado colaborativo *Team Based Learning*, detalhada na Seção 2.6.2. A aplicação da metodologia em questão foi fundamentada na publicação de Parmelee *et al.* (2012).

Todas as aulas foram compostas de estudo prévio, avaliação individual, avaliação em equipe e aplicação prática, sendo a última um desafio de programação pertinente ao conteúdo da aula. O sistema de apoio à realização de maratonas de programação PC^2, apresentado na Seção 2.7.1, foi empregado como juiz automático, auxiliando na correção dos exercícios submetidos e fornecendo *feedback* instantâneo sobre a corretude dos códigos submetidos. Além disso, solicitou-se que cada equipe fizesse um cadastro na plataforma de maneira em que as pontuações, por equipe, pudessem ser exibidas no projetor durante a etapa de aplicação prática. As melhores equipes não foram contempladas com qualquer tipo de pontuação extra.

O sistema de apoio de maratona de programação foi implantando em um *cluster* do Laboratório de Sistemas Distribuído e Programação Concorrente (LaSDPC), e fez-se acesso remoto ao mesmo por *SSH* e por interface *web*. Produziu-se um material de apoio contendo tutoriais de instalação, implantação e uso referente ao sistema PC^2, devido a sua escassa documentação e lições aprendidas com seu uso. O referido material pode ser encontrado em <http://bit.do/material_de_apoio>.

Em cada aula, na etapa de avaliação individual foi aplicada uma prova de 10 questões de múltipla escolha com 4 alternativas, impressa em papel. Na etapa de avaliação em equipe, uma ficha impressa em papel cartão foi utilizada, cuja estrutura pode ser vista na Figura 15. Na etapa de aplicação prática, um documento contendo descrição do desafio de programação a ser implementado foi divulgada por meio do sistema *Moodle* e exibido no projetor durante a porção da aula correspondente a realização desta atividade.

Figura 15 – Estrutura da ficha de respostas do TBL



Além disso, os materiais relacionados tais como descrições e códigos-fonte podem ser vistos no link <http://bit.do/desafios_pc_2018>. Os desafios de programação utilizados foram:

- Encontrar o mínimo (*Pthreads*);
- Multiplicação de matrizes (*Pthreads*);
- Encontrar números primos (*OpenMP*);
- Multiplicação de matriz e vetor (*MPI*);
- Calcular mediana das colunas de uma matriz (*MPI*);
- Contar a quantidade de elementos maiores do que uma entrada (*MPI*);
- Multiplicação de matrizes (*MPI*);
- Multiplicação de um escalar por vetor (*CUDA*);
- Multiplicação de matrizes (*CUDA*).

Finalmente, foi elaborado um questionário qualitativo, afim de avaliar a satisfação dos alunos em função da qualidade das aulas e conteúdo ministrado. Este questionário encontra-se disponível em <http://tiny.cc/qq1>.

Planejamento de Curso

Segue, no Quadro 14, a relação das aulas e conteúdo ministrados. As linhas marcadas em azul constituem aulas consideradas para a consolidação deste experimento, que representam, efetivamente, as ocasiões nas quais foram empregados desafios de programação paralela.

Quadro 14 – Planejamento de curso: primeiro experimento

Aula	Conteúdo
1	Apresentação da Disciplina / Introdução à Programação Concorrente / Conceitos essenciais
2	Revisão de Arquiteturas Paralelas
3	Desenv. de Aplic. Concorrentes: problemas com foco na modelagem/avaliação de desempenho
4	Desenv. de Aplic. Concorrentes: Memória Compartilhada (*PThreads*)
5	Desenv. de Aplic. Concorrentes: Memória Compartilhada (*OpenMP*)
6	Desenv. de Aplic. Concorrentes: Passagem de Mensagens (*OpenMPI*)
7	Desenv. de Aplic. Concorrentes: *OpenMP* e *OpenMPI*
8	**1a. Avaliação Bimestral**
9	Aplicação Prática de *MPI* + *OpenMP*
10	Teste de Software para Programas Concorrentes
11	Desenv. de Aplic. Concorrentes: Processadores Heterogêneos (GPU/*CUDA*)
12	Desenv. de Aplic. Concorrentes: Processadores Heterogêneos (GPU/*CUDA*)
13	Desenv. de Aplic. Concorrentes: Linguagens Funcionais (*Erlang*)
14	Desenv. de Aplic. Concorrentes: Linguagens Funcionais (*Erlang*)
15	**2a. Avaliação Bimestral**

5.3.3 Execução

Preparação

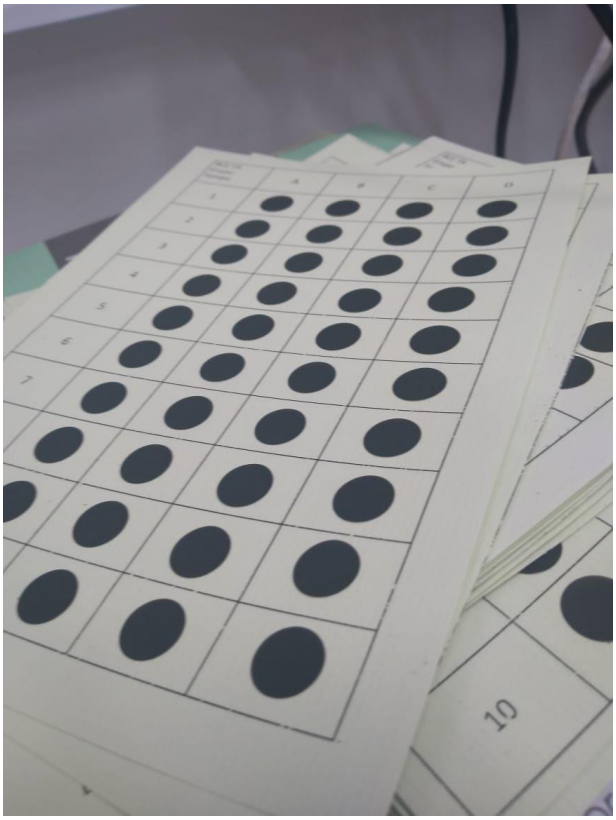
A partir do planejamento do curso, foram selecionados materiais para estudo prévio dos alunos referentes ao conteúdo de cada aula, o que constitui na etapa de preparo ou estudo prévio do *TBL* (PARMELEE *et al.*, 2012).

Antes de cada aula, foram selecionados desafios de programação que abordassem o conteúdo previsto para a ocasião. A descrição de cada desafio foi redigida, além de códigos-fontes (implementação sequencial do problema) para auxiliar os alunos na implementação paralela.

Além disso, foram elaborados testes de múltipla escolha correspondentes às etapas de garantia de preparo (avaliações) individual e em equipe. Os testes individuais tratavam-se de avaliações impressas e nominais, redigidas e impressas antes de cada aula. Os testes em equipe constituíam em uma ficha impressa em papel cartão contendo as mesmas questões e alternativas, cobertas inicialmente por um adesivo, conforme pode ser visto na Figura 16. A utilização desse

material será descrito posteriormente.

Figura 16 – Ficha de respostas utilizada no TBL



Operação

O experimento foi realizado durante o horário das aulas de cada uma das turmas, que tinha duração de 2:30 horas (3 créditos). Em função das etapas do TBL, o tempo de aula foi dividido da seguinte forma.

- 15 - 20 minutos : Garantia de preparo individual;
- 30 minutos : Garantia de preparo em equipe:
- 15 - 20 minutos: Discussão sobre a atividade;
- 45 - 60 minutos: Atividade prática.

O tempo restante era aplicado na organização da turma, visto que a composição dos grupos variava entre as aulas, e dividido entre as etapas, caso o planejado não fosse suficiente.

Os alunos eram organizados em equipes de tamanho variado, em função da quantidade de alunos nas turmas. A composição dos grupos, seguindo as recomendações de Parmelee *et al.* (2012) era alterado entre aulas.

A etapa de garantia de preparo individual era realizada logo no início das aulas, com cada um dos alunos respondendo à um teste de múltipla escolha individual enquanto sentados em fila, em uma organização tradicional de sala de aula. Posteriormente, os alunos eram organizados em grupos, definidos de maneira randômica, e agrupados em pontos distintos da sala.

Na etapa de garantia de preparo em equipe, os alunos respondiam ao mesmo teste da etapa anterior, mas agora em grupos, em que cada equipe possuía uma única folha de resposta, similar à apresentada na Figura 16. O processo de resposta da avaliação equipe constituía na remoção dos adesivos da folha de respostas pelos alunos do grupo. Caso a equipe removesse um adesivo correspondente a alternativa correta (representada pelo símbolo ☺), ela recebia a pontuação máxima daquela questão. Caso a equipe removesse mais adesivos, havia um decréscimo de 25% da nota da questão por resposta incorreta (representada, na ficha de resposta, pelo símbolo ⦸.

5.3.4 Descrição dos Resultados

Aprendizado Técnico

Em um primeiro momento, elencamos os resultados referentes ao aprendizado teórico do conteúdo ministrado. Foram aplicados 7 questionários contendo 10 questões teóricas cada, referentes ao conteúdo da aula .

A primeira etapa do TBL, garantia de preparo individual, reflete o conhecimento teórico do indivíduo após o estudo prévio do conteúdo proposto em ocasião anterior à aula. Cada teste individual foi valorado em 10 pontos e era composto de 10 questões. Cada questão, portanto, valia 1 ponto. Os alunos eram autorizados a distribuir a pontuação de cada questão entre as 4 alternativas possíveis. Desta forma, o aluno que atribuísse o total de pontos na alternativa correta recebia 1 ponto. Caso os pontos fossem atribuídos nas alternativas incorretas, havia um decréscimo de 25% do valor da questão por alternativa incorreta. A proporção da nota da atividade atribuída ao teste individual era de 40%.

A Figura 17 representa a distribuição das médias das pontuações obtidas pelos participantes na primeira etapa do TBL.

A segunda etapa do TBL, garantia de preparo em equipe, representa o conhecimento do aluno em um grupo, consistindo efetivamente, na síntese dos conhecimentos teóricos da equipe e indicando o aprendizado derivado da metodologia em questão. A atribuição de pontuação para cada atividade é similar à etapa anterior, com adição de uma ficha de respostas por equipe, cujo funcionamento foi explicitado na Subseção 5.3.3. A percentagem da nota atribuída a esta etapa era de 18% em relação ao valor total da atividade.

A Figura 18 representa a distribuição das médias das pontuações obtidas pelos participantes na segunda etapa do TBL.

Figura 17 – Histograma das pontuações obtidas na etapa de garantia de preparo individual

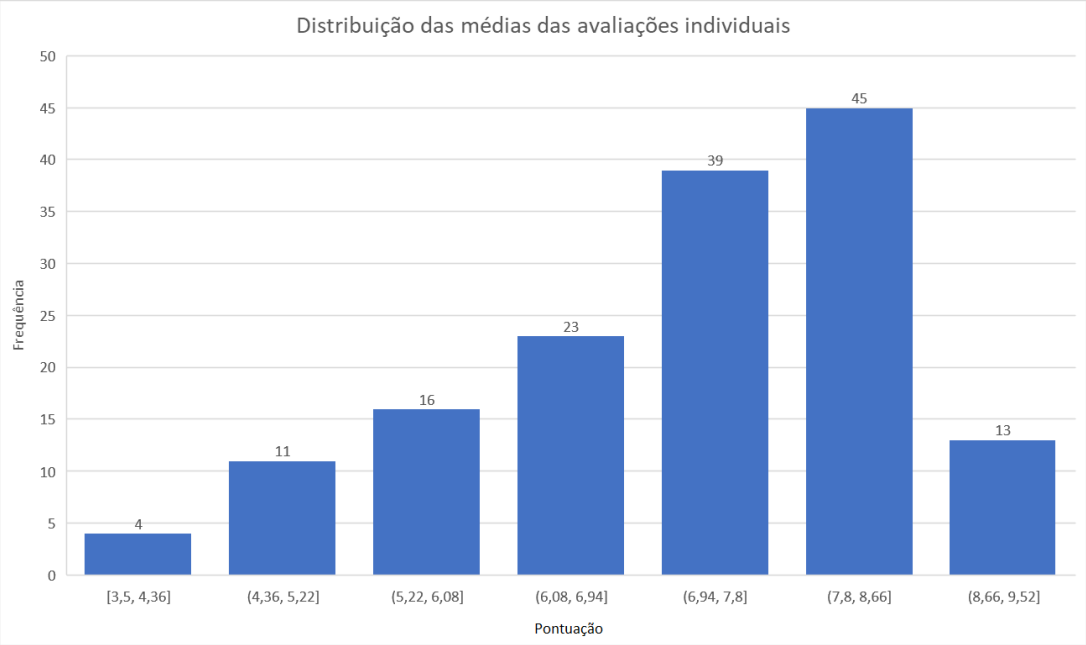
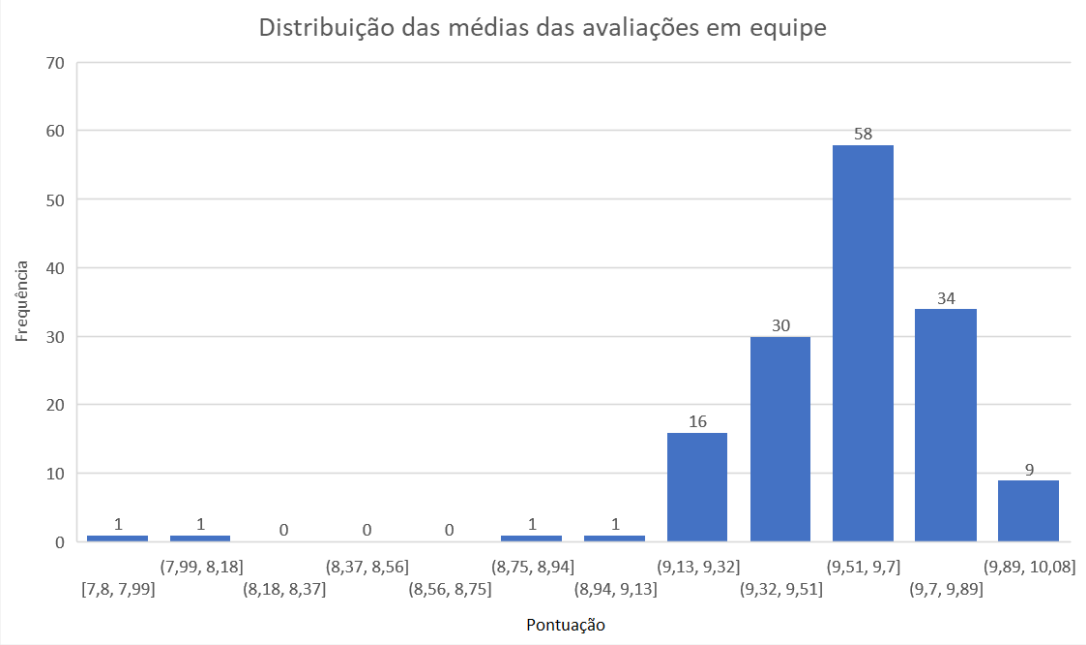


Figura 18 – Histograma das pontuações obtidas na etapa de garantia de preparo em equipe



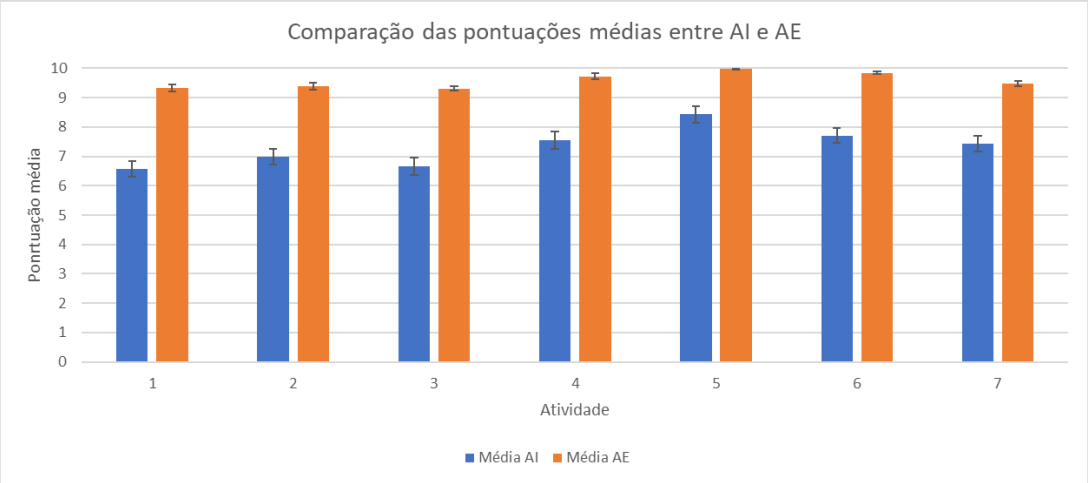
A Tabela 9 apresenta as médias das etapas de garantia de preparo individual (AI) e em grupo (AE) por atividade (AT).

Tabela 9 – Médias individuais e em grupo por atividade

	AT01	AT02	AT03	AT04	AT05	AT06	AT07	Geral
AI	6,6	7,0	6,7	7,5	8,4	7,7	7,4	7,3
AE	9,3	9,4	9,3	9,7	10,0	9,8	9,5	9,6

A Figura 19 ilustra a evolução da pontuação média dos alunos para cada atividade.

Figura 19 – Evolução das pontuações médias por atividade



Verifica-se uma pontuação média de 7,3 da turma na etapa de garantia de preparo individual (AI) e de 9,6 na etapa de garantia de preparo em equipe (AE), demonstrando uma evolução considerável do desempenho em nível de absorção de conhecimento teórico dos alunos entre as etapas do TBL. As barras de erros presentes no gráfico representam o intervalo de confiança normal de 95%, processo que é replicado para os demais gráficos de barra.

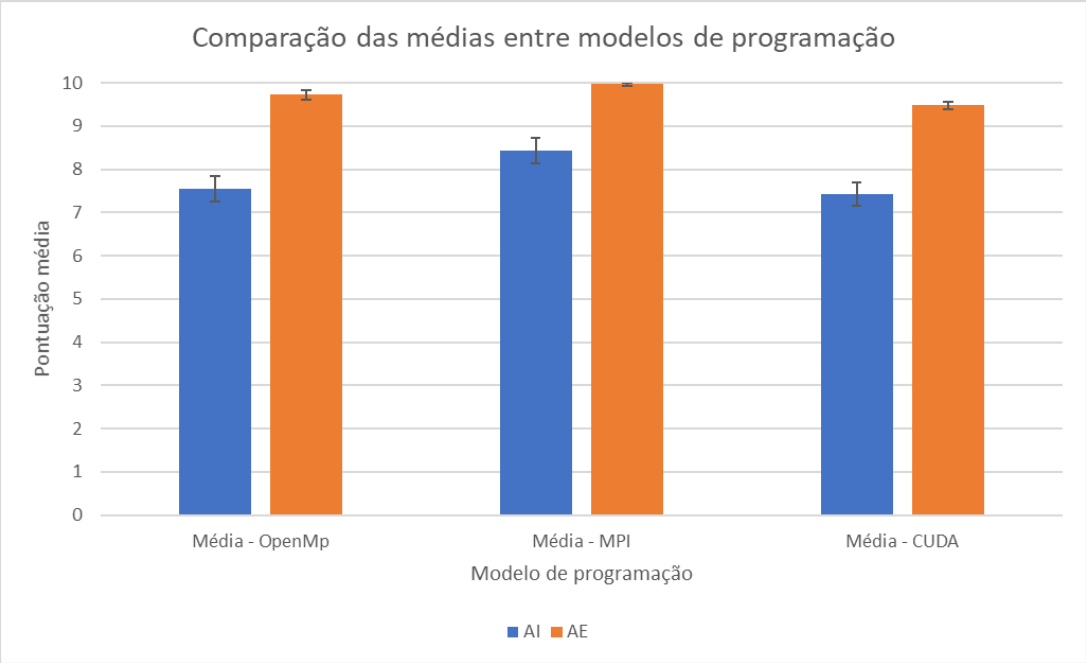
Além disso, podemos avaliar o aproveitamento dos participantes do experimento por modelo de programação. A Tabela 10 apresenta as médias das etapas de garantia de preparo individual (AI) e em grupo (AE) por modelo de programação.

Tabela 10 – Médias individuais e em grupo por modelo de programação

	OpenMP	MPI	CUDA
AI	7,5	8,4	7,4
AE	9,7	10,0	9,5

A Figura 20 ilustra a evolução da pontuação média dos alunos por modelo de programação.

Figura 20 – Evolução das pontuações médias por modelo de programação



Podemos perceber que, para os conteúdos de todos os modelos considerados, houve melhora considerável entre as etapas de garantia de preparo individual e em equipe. Além disso, verifica-se que o aproveitamento médio foi superior nas atividades relacionadas ao modelo de programação MPI e pior em CUDA.

A etapa final do TBL consistia na implementação, em equipe, de uma atividade prática de programação paralela. Essas atividades foram valoradas em 10 pontos cada, atribuídos em função dos critérios estabelecidos na Seção 5.2. A proporção da nota atribuída a esta etapa era de 42% do valor total de cada atividade.

Para a consolidação deste experimento, foram consideradas 6 atividades práticas, elencadas em função do conteúdo abordado no Quadro 14.

A Tabela 11 contém as médias de qualidade de código por atividade prática (AP) realizada
.

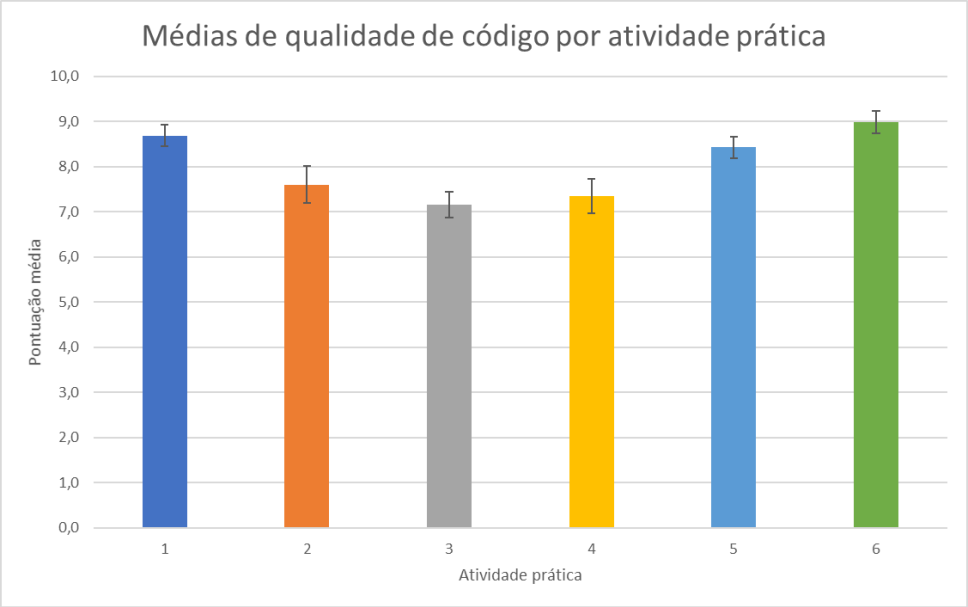
Tabela 11 – Médias de qualidade de código por atividade prática

AP01	AP02	AP03	AP04	AP05	AP06	Geral
8,7	7,6	7,2	7,3	8,4	9,0	8,0

A Figura 20 representa as médias de qualidade de código desenvolvido por atividade prática.

Verifica-se que, de uma forma geral os alunos foram capazes de desenvolver códigos paralelos corretos e com qualidade, apresentando aproveitamento médio de 80%, um mínimo de

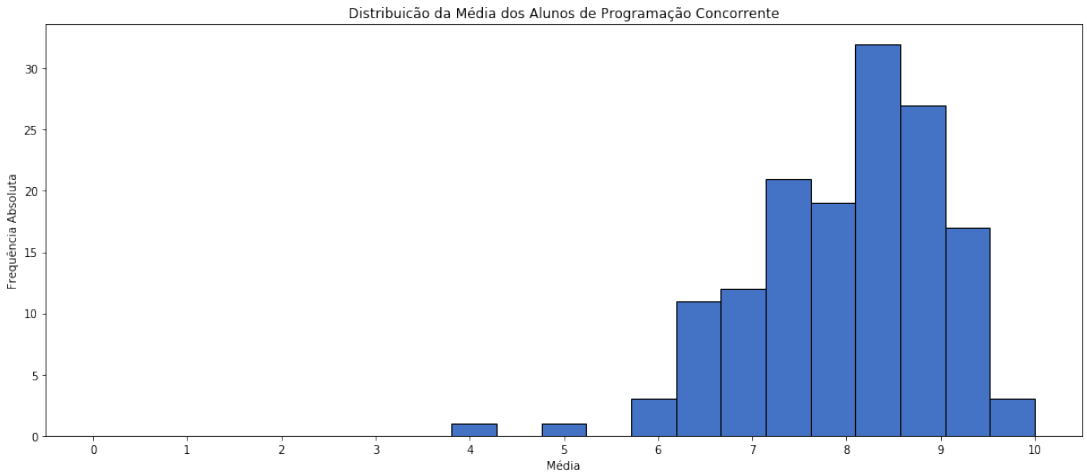
Figura 21 – Pontuações médias de qualidade de código por atividade prática



72% e um máximo de de 90%.

De maneira complementar, a Figura 22 ilustra a distribuição da pontuação obtida pelos participantes em função da qualidade dos códigos-fonte desenvolvidos.

Figura 22 – Histograma das pontuações para qualidade de código desenvolvido



Verifica-se que a maior frequência de pontuações obtidas variou de 8 a 9 pontos, indicando que a maioria dos códigos desenvolvidos apresentou qualidade.

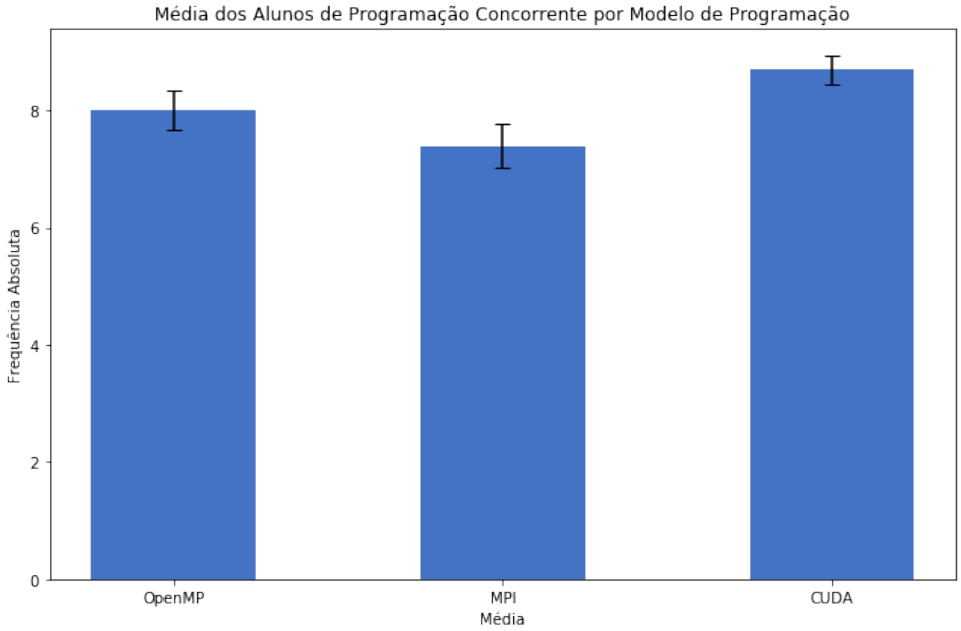
Finalmente, a Tabela 12 descreve as médias obtidas pelos participantes do experimento em critério de qualidade de código, agrupadas por modelo de programação.

Tabela 12 – Médias de qualidade de código por modelo de programação

OpenMP	MPI	CUDA
8,0	7,4	8,7

De forma análoga, a Figura 23 ilustra as médias das pontuações em qualidade de código por modelo de programação.

Figura 23 – Médias de qualidade de código por modelo de programação



É possível perceber que as médias obtidas foram, em geral, satisfatórias, representando um aproveitamento médio mínimo de mais de 70% e máximo de 87%. Adicionalmente, verifica-se que os códigos desenvolvidos para CUDA foram superiores, em relação aos critérios estabelecidos para a consolidação da pontuação de qualidade de código. Entretanto, os códigos desenvolvidos para *MPI* apresentaram qualidade inferior em relação aos demais.

Avaliação qualitativa

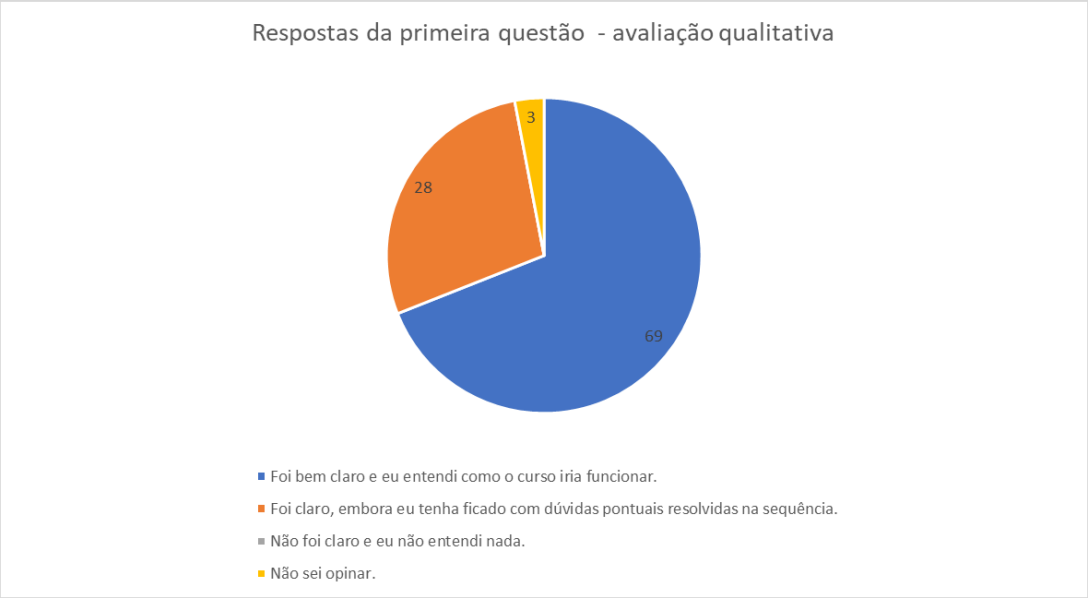
Ao fim do experimento, foi divulgado um questionário *online*, por meio do *Moodle*, cuja resposta era anônima e optativa.

Houve um total de 100 respostas ao formulário, o que correspondente à 66% dos participantes. As respostas podem ser encontradas integralmente no link <http://bit.do/exp1_qualitativo>.

A seguir, sintetizamos as respostas para as perguntas consideradas mais relevantes do teste qualitativo. A análise das respostas obtidas será feita posteriormente.

1) O planejamento do curso foi suficientemente claro até o início da segunda aula?

Figura 24 – Respostas da 1ª questão - avaliação qualitativa

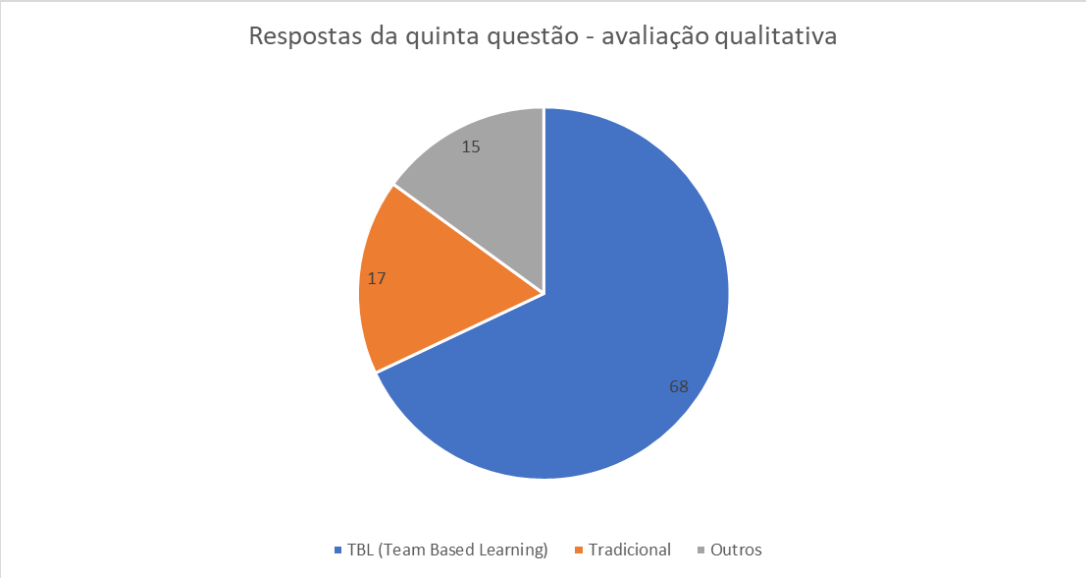


3) Foi possível compreender o funcionamento do TBL (Team Based Learning), suas etapas e características até o início da segunda aula?

Figura 25 – Respostas da 3ª questão - avaliação qualitativa

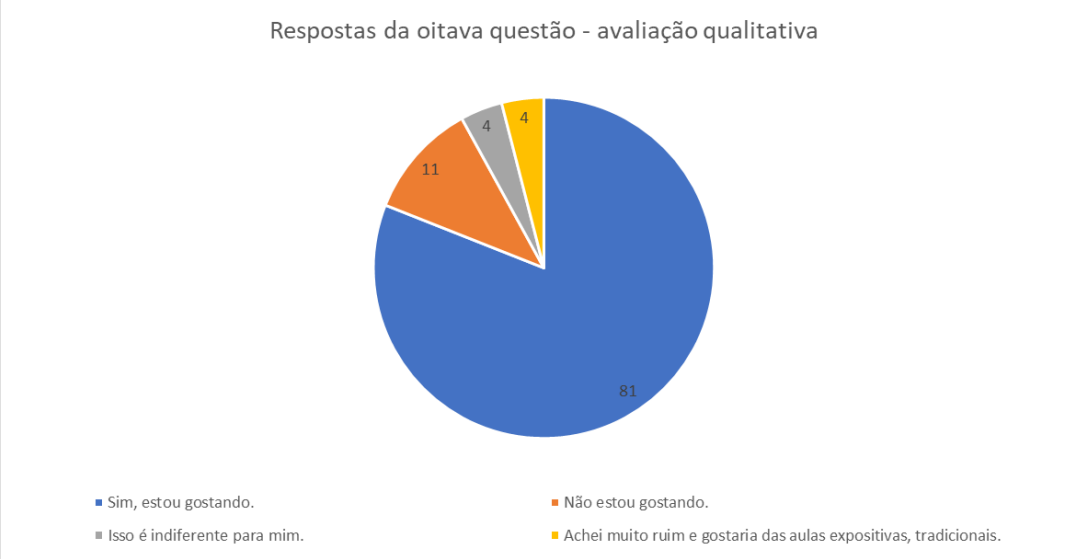
5) Qual método de ensino/aprendizado você acredita ser melhor?

Figura 26 – Respostas da 5ª questão - avaliação qualitativa



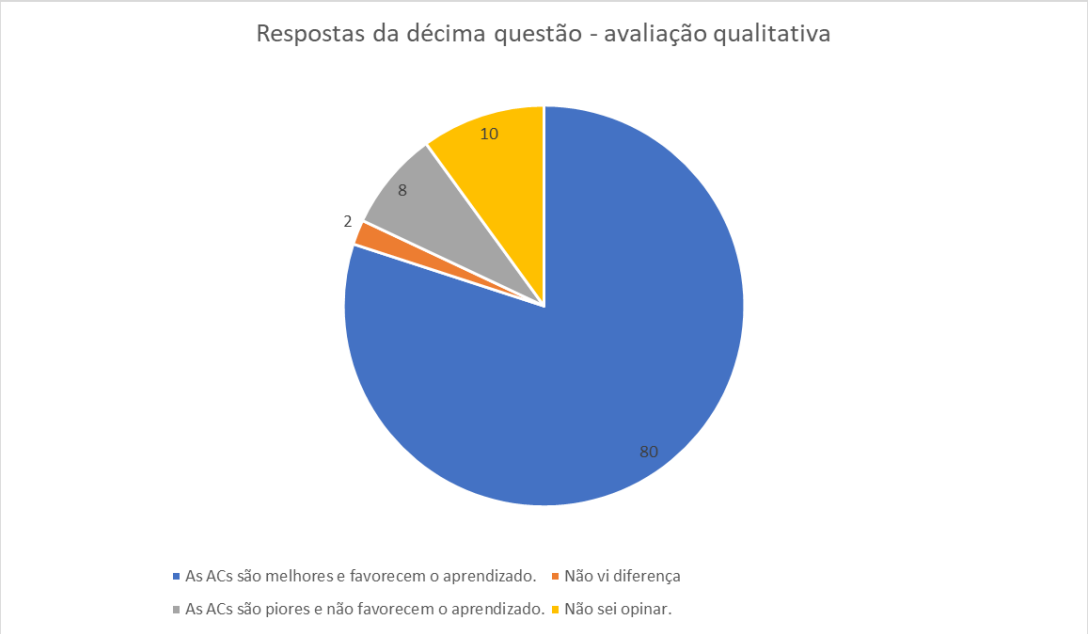
8) Você está gostando das aulas com TBL?

Figura 27 – Respostas da 8ª questão - avaliação qualitativa



10) Qual sua opinião sobre a aplicação de avaliações contínuas (ACs)?

Figura 28 – Respostas da 10ª questão - avaliação qualitativa

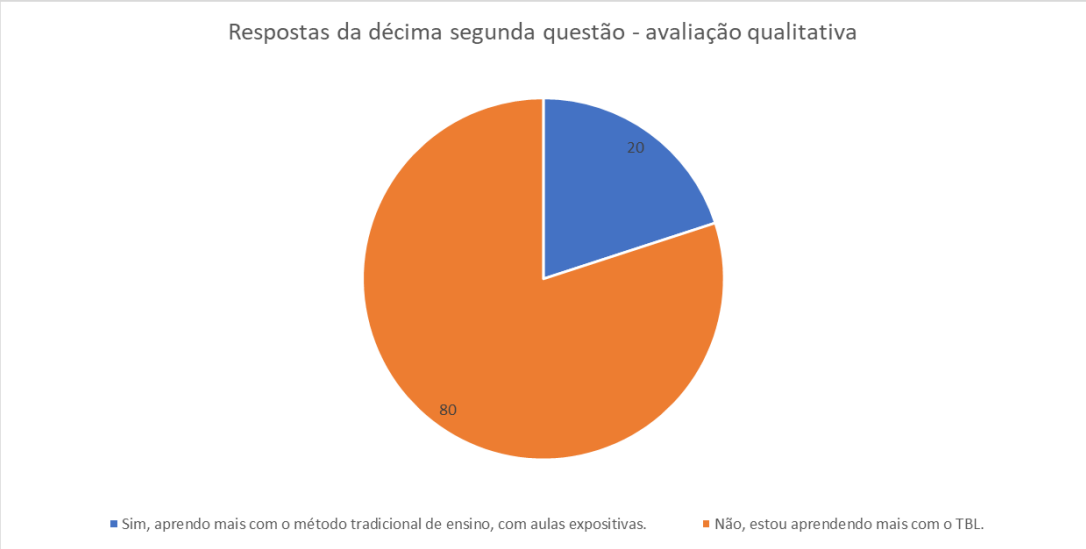


11) Em relação às quatro etapas do TBL, qual você acredita que melhor favorece seu aprendizado?

Figura 29 – Respostas da 11ª questão - avaliação qualitativa

12) Você gostaria que o curso voltasse a ser ensinado pelo método tradicional de ensino, com aulas expositivas?

Figura 30 – Respostas da 12ª questão - avaliação qualitativa



13) Dê suas sugestões, críticas e/ou elogios sobre o curso em andamento.

Apresenta-se aqui uma amostra dos comentários providos pelos alunos participantes, selecionados de forma a representar, ainda que de forma sucinta, a opinião geral dos estudantes.

- "Parabéns pelo esforço para tornar a disciplina mais agradável e, ao mesmo tempo, que garanta o aprendizado por parte dos alunos. (. . .) "
- "Acredito que o método TBL é muito superior às aulas expositivas, por ser uma forma mais dinâmica de aprendizado, muito menos maçante e que gera uma forma mais contínua de conhecimento do que picos de atividade perto de datas importantes como provas";
- "De modo geral a matéria está sendo muito interessante e engajante para mim. As aulas estão com uma carga de conteúdo correta, e não senti até agora que nenhuma das aulas tinham mais matéria do que eu conseguia processar, ou pouca matéria o suficiente para que se torne entediante".
- "Em geral, acredito que esta matéria está sendo uma das melhores (talvez a melhor) matéria que eu tive na universidade, pois além de ter noção da importância dela na minha futura carreira, sinto que todos os conceitos estão sendo fixados de maneira que estarei confiante de usá-los no futuro, e que tenho o necessário para dominá-los com a prática".
- "Eu acho que uma ou outra aula expositiva, que seja para revisar os conteúdos aprendidos e ajudar a fixar os conceitos seria bastante útil. Acho importante também que o conteúdo para estudo prévio seja bem claro e não muito grande, para não pesar tanto nos nossos estudos e deixar o conteúdo bem direcionado para a prova".
- "Achei uma abordagem nova muito interessante porém extremamente exigente quando colocada junto do restante das matérias no sistema expositivo, portanto acho valido manter

este método porém dando maior importância para o fato de que os alunos tem mais matérias para estudar na semana (. . .).”

- “Minha única crítica é em relação as aplicações práticas, nos grupos em que participei, o tempo se mostrou ser meio curto. ” "Apesar de estar gostando do método TBL, pelas características citadas anteriormente, acredito que em disciplinas como estas, que envolvem programação, as aulas expositivas são muito importantes.”

5.3.5 Análise

A partir dos resultados obtidos e expostos previamente, visa-se, nesta etapa estabelecer as respostas para as questões de pesquisa elencadas na Subseção 5.3.2 e demais análises pertinentes a partir dos produtos deste experimento.

[QP1] Os alunos foram capazes de aprender o conteúdo teórico de *OpenMP*, *MPI* e *CUDA*?

Percebeu-se, por meio da análise das Figuras 17 e 18 que os alunos obtiverem maior frequência de pontuações positivas, tanto para a etapa de garantia de preparo individual quanto para garantia de preparo em equipe.

Além disso, por meio da Figura 19 é possível perceber que, em média, os alunos obtiveram bom desempenho em ambas etapas do TBL consideradas, apresentando médias de 7,3 em atividades individuais e de 9,6 em atividades em grupo, o que representa uma evolução considerável, em critérios de absorção de conhecimento, entre as duas etapas. Adicionalmente, percebe-se, analisando a Figura 20, que os estudantes apresentaram bom desempenho para todos os modelos de programação.

Portanto, é possível afirmar que os alunos obtiveram bom desempenho em ambas etapas, garantia de preparo individual quanto para garantia de preparo em equipe, e foram capazes de aprender de maneira satisfatória o conteúdo teórico ministrado no contexto deste experimento.

[QP2] Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP*, *MPI* e *CUDA* corretas e com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho?

Por meio da análise da Figura 21 é possível perceber que os alunos obtiveram boas pontuações referentes a qualidade de código em todas os desafios de programação propostos, ilustradas por um aproveitamento médio de 80%, um mínimo de 72% e um máximo de de 90%.

Além disso, verifica-se, com base na Figura 22, que as pontuações obtidas mais frequentemente figuravam de 80% a 90% do valor da questão, representando que, em geral, os alunos desenvolveram códigos paralelos de qualidade, considerando os critérios considerados.

Finalmente, considerando a Figura 23, é possível perceber que os estudantes desenvolveram programas paralelos com qualidade para todos os modelos de programação considerados,

ainda que com aproveitamento superior em CUDA e inferior em MPI.

Desta forma, é possível afirmar que os participantes deste experimento foram capazes de desenvolver códigos paralelos com corretos, com qualidade e desempenho.

[QP3] Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?

A Figura 24 indica que houve a compreensão dos alunos ao planejamento e estrutura de curso proposta logo no início do experimento, explicitando que, na opinião dos presentes, as aulas foram propostas com um planejamento claro e bem definido.

É possível perceber, por meio da análise da Figura 25, que a maior parte da amostra compreendeu a metodologia, seus objetivos e particularidades.

Por meio da interpretação dos resultados ilustrados pela Figura 26 que houve uma preferência clara da maioria dos alunos pela metodologia TBL, em relação, principalmente, ao método tradicional, que historicamente havia sendo aplicado nesta disciplina.

A Figura 27 ilustra que, em geral, os alunos ficaram satisfeitos com a metodologia aplicada.

Analisando a Figura 28 é possível perceber que a maior parte dos participantes do experimentos aprovaram a aplicação de avaliações contínuas, fundamentadas na metodologia do TBL e com o uso de desafios de programação.

O conteúdo da Figura 29 ilustra que, na sua maioria, os estudantes avaliaram o estudo prévio do conteúdo como a etapa mais importante do TBL, seguida pela garantia de preparo em equipe.

Por sua vez, a Figura 30 indica que a maior parte dos alunos preferiram aprender com o TBL e não gostariam que o conteúdo voltasse a ser ministrado no formato tradicional.

Finalmente, a análise dos comentários providos pelos participantes, no contexto da questão 13 do questionário qualitativo, indica a satisfação dos alunos com o método, conteúdo e didática empregada no processo pedagógico. Entretanto, houveram sucessivas críticas ao tempo para desenvolverem as atividades, seja ela o estudo prévio do conteúdo antes da aula ou o desenvolvimento das tarefas em sala.

5.4 Experimento II: Curso de Difusão Cultural de Introdução ao *OpenMP* para alunos iniciantes em computação

Esse experimento foi conduzido por outro integrante do grupo de pesquisa, com auxílio de um professor doutor e outro aluno. O objetivo geral do presente experimento é analisar o uso

de desafios de programação, como mecanismo de avaliação e complemento de aprendizagem no contexto de um curso de extensão, ministrado com base na metodologia de ensino tradicional.

5.4.1 Escopo

Objetivo

O objetivo deste experimento foi avaliar a capacidade de alunos ingressantes em computação de compreenderem os conceitos de paralelismo e desenvolverem aplicações paralelas com qualidade usando o modelo de programação *OpenMP* e linguagem de programação C.

Objeto

Foi definido como objeto de estudo a proposta do ensino de programação paralela para discentes matriculados no primeiro bimestre de graduação em cursos relacionados à computação, permitindo que os mesmo possam desenvolver códigos paralelos desde o início da graduação.

Foco qualitativo

Estabeleceu-se como foco qualitativo deste experimento a atividade de mensurar o desempenho dos estudantes quando submetidos a desafios de programação em uma metodologia tradicional de ensino. Neste contexto, os desafios foram agrupados em uma maratona de programação, realizada no última dia de aula. Além disso, busca-se averiguar os graus de atenção, relevância, confiança e satisfação, segundo o modelo *ARCS*, dos alunos referente ao conhecimento adquirido aplicando a Pesquisa de Interesse de Curso, definida em Keller (2010).

Perspectiva

Considerando o contexto de alunos ingressantes na graduação e baseando-se no modelo de ensino tradicional, a perspectiva deste experimento considerou o ambiente real de sala de aula. Além disso, por tratar-se de um curso com aspecto prático e, principalmente, para a realização de uma maratona, o ambiente ideal para a condução do experimento é um laboratório de computação contendo computadores individuais disponíveis para os alunos.

Contexto

Em função da perspectiva proposta, esse estudo experimental foi realizado no formato de curso de difusão cultural nas dependências de um laboratório de computação no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo. O referido curso foi estruturado com carga horária total de 18 horas dividas em 6 aulas, de 3 horas cada. Na ocasião da última aula, foi realizada uma maratona de programação paralela.

5.4.2 Planejamento

Definição das questões de pesquisa

Dentro do contexto deste experimento, podemos elencar as seguintes questões de pesquisa:

QP1 Os alunos foram capazes de absorver o conteúdo teórico de *OpenMP*? (avaliação dos pré e pós testes);

QP2 Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP* corretas e com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho?;

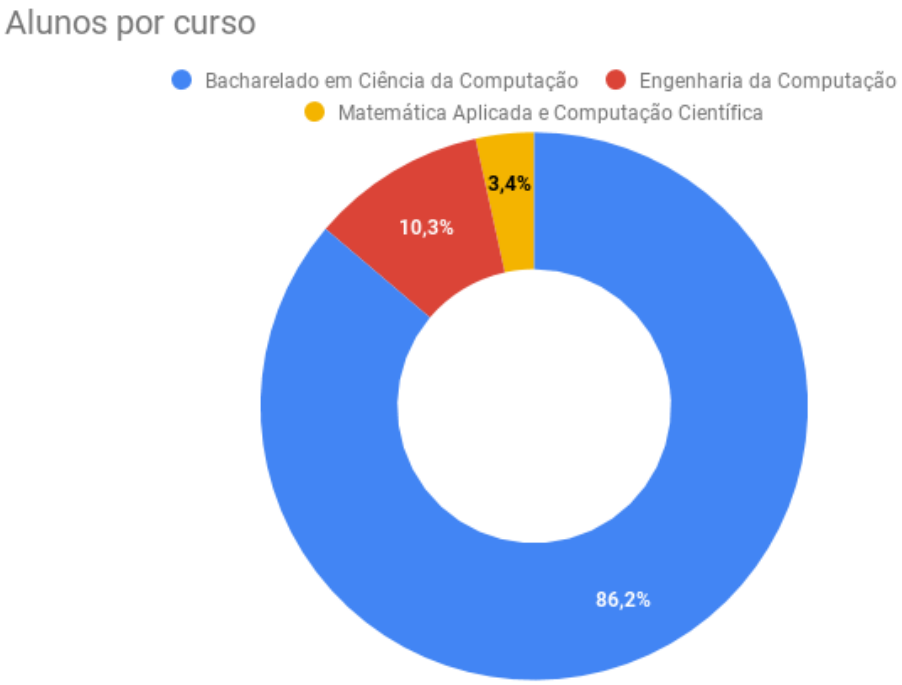
QP3 Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?.

Seleção de sujeitos

Dada a natureza de atividade de extensão e que permitia a matrícula livre e restrita para a comunidade, os pesquisadores não exerceram influência sobre a seleção da amostra. Desta forma, os sujeitos deste experimento foram selecionados de forma arbitrária, a partir da matrícula no curso.

Os participantes deste experimento podem ser agrupados de acordo com a Figura 31.

Figura 31 – Distribuição dos alunos do curso de OpenMP I



Descrição de instrumentação

O curso foi estruturado no formato de atividade de extensão, não obrigatório, cujas vagas foram abertas à comunidade. Houve preferência pela inscrição de alunos de início de graduação

em computação (1º ano), totalizando 29 alunos participantes em todas tarefas. O curso teve duração total de 18 horas, divididas entre 6 tópicos/aulas de 3 horas cada. A metodologia de ensino/aprendizado aplicada neste curso foi tradicional, com uso de desafios de programação como forma de fixação e prática dos conceitos. Foi utilizada maratona de programação e sistema de apoio a realização de maratonas de programação *Mooshak 2* (descrito na Seção 2.7.1 e disponível em <https://mooshak2.dcc.fc.up.pt/>) no última dia de curso.

O sistema *Mooshak* foi implantado em uma máquina virtual da *Google Cloud* e os alunos puderam acessa-lo por meio de uma interface *web*, um serviço disponibilizado pela própria aplicação. O processo de instalação, configuração e uso do *Mooshak*, além da configuração da máquina virtual na plataforma *Google Cloud* podem ser vistos no *link* <http://bit.do/material_de_apoio>.

Os desafios propostos para este curso foram:

- Ex 1 - Encontrar o mínimo
- Ex 2 - Somar números ímpares em um intervalo
- Ex 3 - Ordenação dos elementos de um vetor
- Ex 4 - Multiplicação de Matrizes

O material associado a estes desafios encontra-se disponível em <http://bit.do/desafios_omp1>.

Planejamento de curso

Segue, no Quadro 15, a relação entre os tópicos ministrados e objetivos associados.

5.4.3 Execução

Preparação

Antes da realização deste curso, os desafios foram selecionados, descritos, implementados, em versões sequencial e paralela. Além disso, realizaram-se configurações e testes do sistema *Mooshak* na plataforma *Google cloud*, o que envolveu criação de máquina virtual, configuração de rede e roteamento, definição de endereço IP público e testes do serviço *web* e dos códigos implementados. Conforme dito anteriormente, este procedimento é melhor descrito no material contido em <http://bit.do/material_de_apoio>. As máquinas do laboratório no qual ocorreu a maratona foram testadas para garantir a capacidade de execução dos códigos localmente e submissão remota.

Além disso, foram adquiridos prêmios a serem entregues às equipes melhores classificadas na maratona de programação.

Quadro 15 – Planejamento de curso: segundo experimento.

Aula	Objetivos
1	Introduzir o pensamento algorítmico Estimular o pensamento paralelo em soluções algorítmicas Iniciar o desenvolvimento de código em linguagem C Ensinar como gerar execuções paralelas de algoritmos em C
2	Introduzir o conceito de variáveis em C Apresentar operadores aritméticos, relacionais e lógicos Criar operações lógicas com os operadores Apresentar a estrutura de decisão `if...else` em linguagem C Utilizar `omp_sections` para obter paralelismo funcional
3	Conhecer o conceito de estruturas de repetição Utilizar a estrutura de laço `for` Utilizar a estrutura de laço `while` Apresentar o paralelismo em laços de repetição
4	Apresentar o conceito de vetores Demonstrar como são usados e estruturados Ensinar como acessar posições em um vetor
5	Apresentar o conceito de modularização e reuso de código Demonstrar o uso de funções em C Passagem de parâmetros de entrada e saída Exercícios práticos com funções destacando aspectos paralelos Paralelismo com `sections` do *OpenMP*
6	Concorrência na escrita e leitura de dados compartilhados Garantindo a semântica das operações sobre dados compartilhados Exercícios práticos com dados compartilhados destacando os conceitos vistos

Operação

Durante o último dia de curso, foi solicitado que os alunos formassem grupos de até 3 integrantes e se reunissem em bancadas de computadores distintas. Exibiu-se a descrição dos desafios em um projetor e também foi disponibilizado um *link* para um documento contendo a descrição dos desafios e códigos sequenciais bases para as implementações.

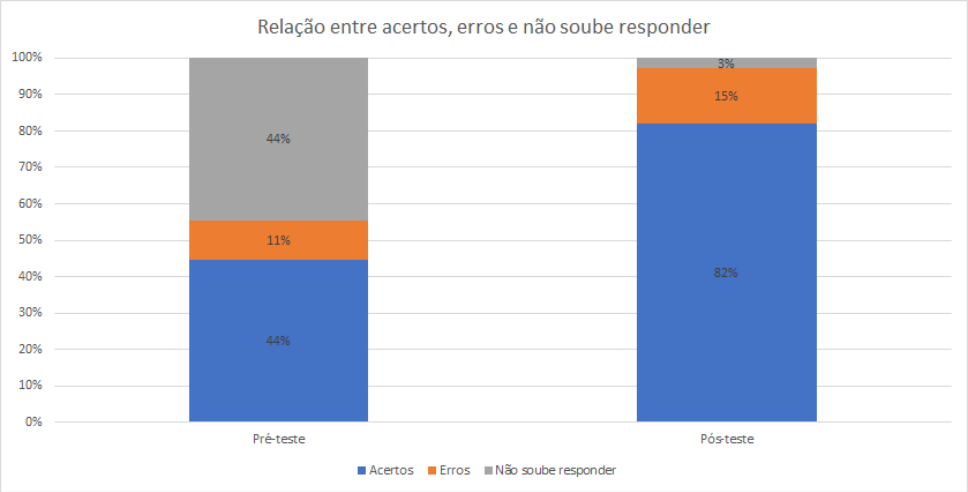
O professor e os dois alunos presentes atuaram respondendo à dúvidas dos participantes e no monitoramento do sistema. Durante a maior parte da aula, foi exibido o placar da maratona em um projetor, no qual as equipes puderam consultar suas classificações, que também fica disponível através do próprio sistema *Mooshak*.

Ao fim da aula, os códigos submetidos ainda foram avaliados manualmente antes de anunciar as equipes vencedoras, que foram então agraciadas com prêmios simbólicos (*mouse*s sem fio e bombons).

5.4.4 Resultados

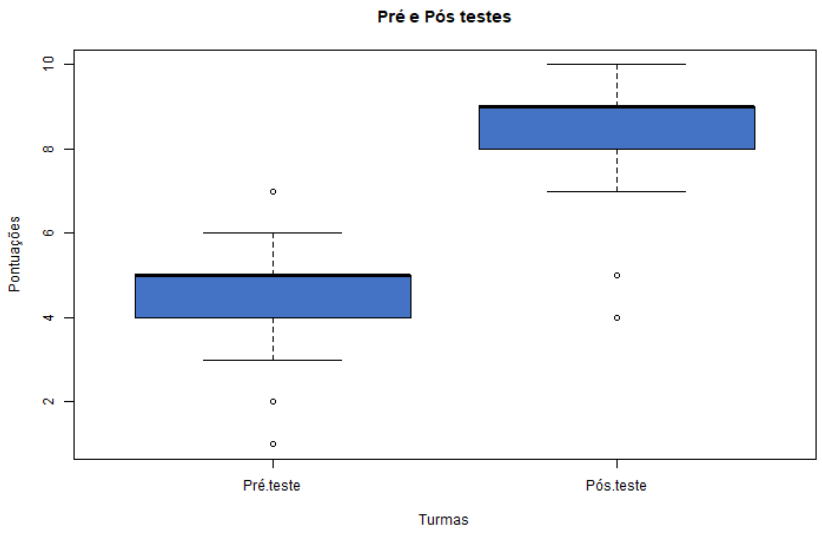
Aplicou-se um pré-teste, na ocasião da primeira aula e um pós-teste na última aula, com objetivo de avaliar a absorção do conhecimento teórico pelos participantes deste experimento. Um total de 29 alunos realizou ambos testes. A Figura 32 representa o percentual de acertos, erros e ocasiões as quais os alunos não sabiam a resposta. É possível verificar uma melhoria de 44% de acertos, no pré-teste, para 82%, no pós-teste.

Figura 32 – Percentagem de acertos, erros e "não sei"do pré-teste e pós-teste



A Figura 33 apresenta a comparação entre as pontuações de ambos testes, em formato de *boxtplot*. É possível verificar uma distinta melhoria nos *scores* obtidos no pós-teste, em detrimento daqueles que haviam sido estabelecidos no pré-teste.

Figura 33 – Comparação entre as pontuações do pré-teste e pós-tese



No intuito de embasar as afirmações acerca do desempenho dos alunos, fizeram-se

algumas análises. Primeiramente, o teste de *Shapiro-Wilk* foi feito para confirmar a normalidade dos dados, com nível de significância de 95%, gerando os resultados que seguem:

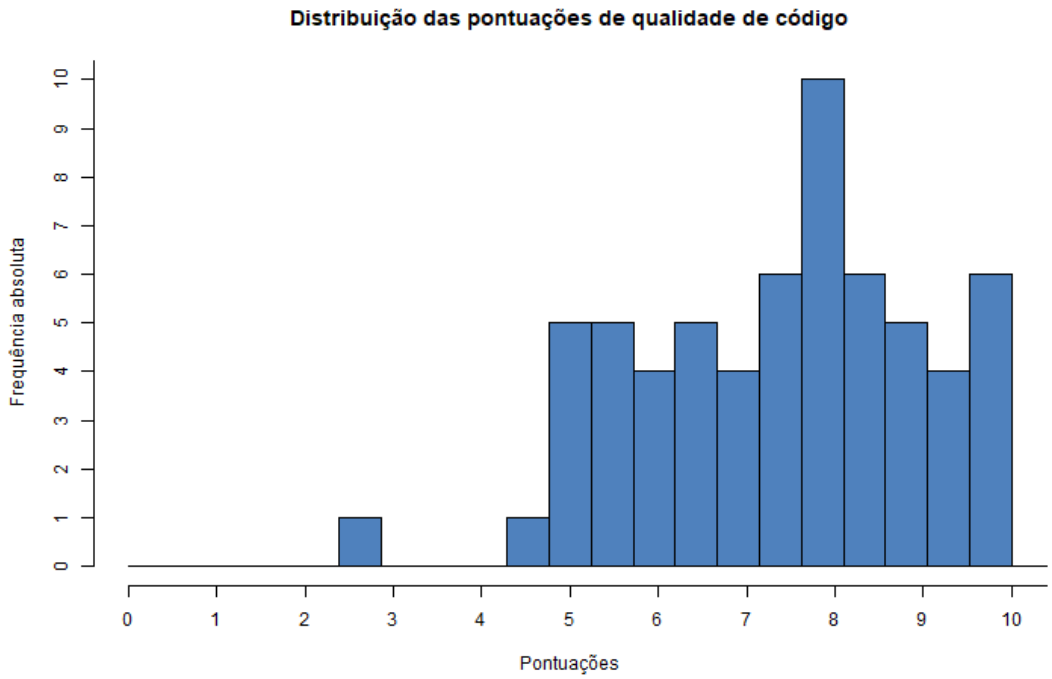
- Pré - teste: W = 0,8223, *p-valor* = 0,0002. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0,8101, *p-valor* = 0,0001. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.

Em razão da não normalidade do conjunto de dados, fez-se o teste de *Wilcoxon* para amostras pareadas, também com 95% de significância, no intuito de avaliar a equivalência (*h0*) ou não (*h1*) das médias obtidas em cada teste, produzindo os seguintes resultados:

- *p-valor* = $2{,}296 \cdot 10^{-6}$, *p-valor* < 0,05. Conclusão: Rejeita-se *h0*, significando que os alunos obtiveram melhores notas, em média, no pós-teste.

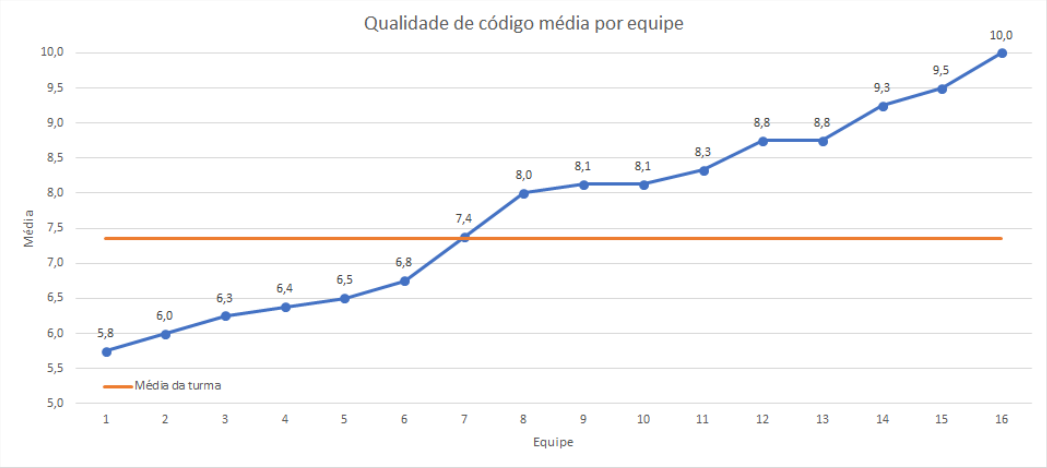
Um outro resultado obtido constitui no aproveitamento em função da qualidade de código, calculada a partir dos parâmetros estabelecidos anteriormente. Primeiramente, a Figura 34 apresenta a distribuição das pontuações obtidas para qualidade de código, considerando apenas os 4 desafios de programação realizados no contexto da última aula do curso. Verifica-se que maior frequência de pontuações figura entre 7 a 8 pontos, significando um bom desempenho da turma, traduzido em uma média da turma de 74% de aproveitamento.

Figura 34 – Distribuição das pontuações obtidas nos desafios de programação



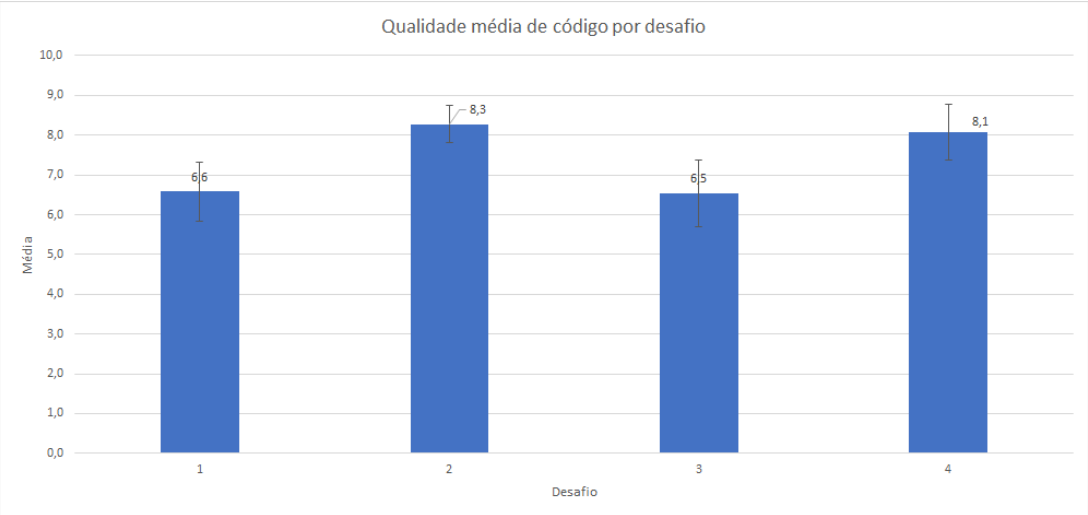
A Figura 35, por sua vez, traz uma visão da pontuação média de qualidade de código por equipe. É possível perceber que 62,5% das equipes conquistaram médias iguais ou superiores do que a média da turma. Além disso, percebe-se que a distância entre as pontuações é bastante reduzida.

Figura 35 – Média de qualidade de código por equipe



Pode-se também avaliar as pontuações médias por desafio realizado, o que é explicitado na Figura 36. Neste caso, verifica-se que, em todos os desafios, a média de qualidade obtida foi superior a 65% da nota válida para este critério.

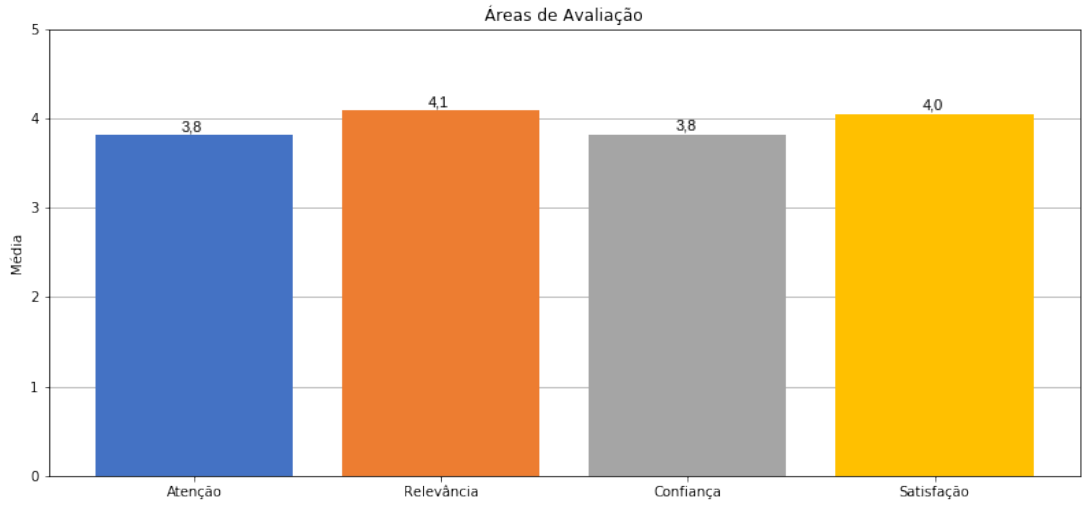
Figura 36 – Média de qualidade de código por desafio



Avaliação qualitativa

Afim de mensurar a satisfação dos participantes deste experimento, foi aplicado um questionário, derivado do *Course Interest Survey* de Keller (2010). A Figura 37 apresenta as médias das pontuações, para cada uma das categorias do modelo *ARCS*. Pode-se perceber que o grau de motivação médio para este experimento foi de 72,5%.

Figura 37 – Avaliação do curso na escala likert, por categoria do modelo *ARCS*



5.4.5 Análise

A seguir, busca-se responder às questões de pesquisa definidas para este experimento através da provisão de dados que sejam capazes de satisfazer as indagações propostas.

[QP1] Os alunos foram capazes de absorver o conteúdo teórico de *OpenMP*?

É possível notar, analisando a Figura 33 e da análise estatística associada, que houve uma melhora significativa no aproveitamento dos participantes deste experimento, representada por uma percentagem de acertos de 82% no pós-teste, em vista dos 44% verificados no pré-teste e da redução do percentual de situações nas quais o aluno não soube responder de 44% para 3%.

Desta forma, é razoável afirmar que houve aprendizado do conteúdo teórico ministrado no curso pelos discentes presentes.

[QP2] Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP* corretas e com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho?

A Figura 35 mostra que a maioria das equipes obteve pontuação superior, em média, do que a média da turma (74% de aproveitamento) nos desafios propostos, enquanto a Figura 36 indica que as pontuações, em geral, foram superiores a 65% do conceito válido para qualidade de código em todos os desafios considerados.

Em vista destes dados, é possível dizer que os estudantes foram capazes de desenvolver códigos corretos e com qualidade, o que foi validado com auxílio do sistema *Mooshak* e de correção manual feita por professores.

[QP3] Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?

Pode-se notar, com base na Figura 37, que para todas as categorias do modelo *ARCS*,

houve pontuação média, na escala *likert*, acima de 60%, caracterizando uma aprovação média do curso de 72,5%.

Desta forma, julga-se factível afirmar que os alunos se sentiram satisfeitos e motivados em relação ao método, conteúdo, didática e recurso educacional empregados, além do uso de desafios de programação e de um ambiente de maratona de programação paralela.

5.5 Experimento III: Curso de Difusão Cultural de Introdução ao *OpenMP* baseado em *Problem Based Learning*

5.5.1 Escopo

Objetivo

O objetivo deste experimento é investigar o processo de ensino-aprendizagem em um curso de programação paralela totalmente baseado em desafios de programação. Deseja-se avaliar o impacto causado pela abordagem no conhecimento técnico e na motivação dos alunos participantes.

Objeto

O objeto de estudo deste experimento é a utilização de uma abordagem baseada em desafios voltada para o ensino de programação paralela associada à maratona de programação paralela para consolidação do conhecimento.

Foco qualitativo

O foco qualitativo deste experimento é avaliar o desempenho dos estudantes na resolução dos desafios escolhidos para aplicação durante o curso e material selecionado para auxílio. Além disso, deseja-se mensurar a motivação dos alunos no curso por meio do modelo ARCS.

Perspectiva

Considerando o aspecto prático dos desafios, a perspectiva é de que o experimento seja executado de forma que forneça todos os recursos necessários para a submissão dos desafios e realização da maratona, portanto, em um laboratório com acesso a internet e computadores individuais possuindo *hardware* e *software* suficientes para a implementação de todos os desafios.

Contexto

O curso foi projetado para um curso de difusão cultural com vagas abertas à comunidade, realizado em junho de 2019 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo em um laboratório de computação que supria as condições necessárias para realização do mesmo. O experimento foi elaborado com carga horária total de 12 horas

distribuídas em 4 aulas de 3 horas cada.

5.5.2 Planejamento

Definição das questões de pesquisa

As seguintes questões de pesquisa foram estabelecidas em razão do objetivo:

QP1 Os alunos foram capazes de absorver o conteúdo teórico de *OpenMP*?

QP2 Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP* com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho?; e

QP3 Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?

Seleção de sujeitos

O curso foi aberto à comunidade, cuja matrícula foi livre e irrestrita para qualquer pessoa interessada. Desta forma, a seleção de sujeitos foi randômica e os pesquisadores não exerceram qualquer influência sobre o processo.

A amostra deste experimento foi composta por um total de 47 estudantes inscritos no curso, de 15 diferentes cursos. A composição dessa amostra, por grau de instrução é ilustrada na Figura 38.

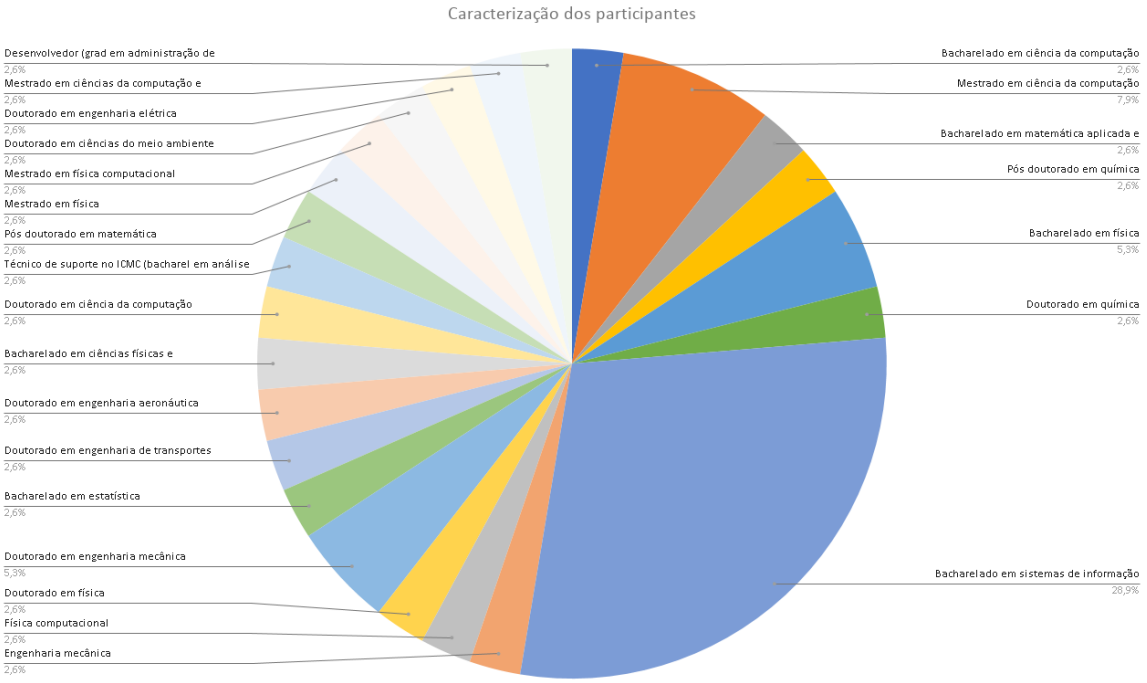
Descrição da instrumentação

O curso foi estruturado no formato de atividade de extensão, não obrigatório, cujas vagas foram abertas à comunidade. Houve preferência pela inscrição de alunos de início de graduação em computação (1º ano), ainda que a turma tenha sido composta de forma mista, totalizando 21 alunos participantes em todas as tarefas.

O curso teve duração total de 12 horas, divididas entre 4 tópicos/aulas de 3 horas cada. A metodologia de ensino/aprendizado aplicada neste curso foi *Problem Based Leaning* (PBL), fundamentada no uso de desafios de programação como elemento balizador do conteúdo, de forma que a teoria é apresentada como recurso para solução dos desafios propostos. Foi utilizada maratona de programação e sistema de apoio à realização de maratonas de programação *Mooshak 2* (descrito na Seção 2.7.1 e disponível em <https://mooshak2.dcc.fc.up.pt/>) em todas as aulas. Além disso, a última aula foi essencialmente composta de uma maratona de programação paralela incluindo premiação simbólica das equipes melhor classificadas.

De forma similar ao experimento anterior, a utilização do sistema *Mooshak* foi feita a partir de uma máquina virtual do *Google cloud*. O material de apoio referente ao processo de instalação e utilização do sistema pode ser visto em <http://bit.do/material_de_apoio>.

Figura 38 – Caracterização dos participantes por grau de instrução



Foram propostos um total de 16 desafios para este curso, cujos códigos sequencial e paralelo encontram-se no *link* <http://bit.do/desafios_omp_2019_2>.

Adicionalmente, também foram redigidos cadernos de desafios de programação paralela, que trata-se de um documento contendo, primeiramente, a descrição dos desafios de programação a serem implementados, acompanhados pelo referencial teórico básico para a resolução das atividades e, por fim, da implementação sequencial dos desafios propostos.

Um documento neste formato era disponibilizado aos alunos em versão impressa e também *online* ao início de cada aula.

O endereço <http://bit.do/caderno_desafios_curso> contém todos os cadernos de programação elaborados para utilização no contexto deste experimento.

Planejamento de curso

Segue abaixo, no Quadro 16, a relação dos tópicos ministrados e conceitos associados.

5.5.3 Execução

Preparação

Em um primeiro momento, fez-se a seleção dos desafios de programação a serem aplicados no curso, que foram, então, descritos, implementados em versões sequencial e paralela. De forma análoga ao experimento anterior, o ambiente no *Google cloud* foi preparado para

Quadro 16 – Planejamento de curso: sexto experimento.

Aula	Objetivos
1	Desafios de programação Diretiva `parallel` Escopo de variáveis (`shared`, `private`, `firstprivate`, `lastprivate`, `threadprivate`, `copyin`)
2	Desafios de programação Diretiva `for` (loops paralelos) Redução (diretiva `reduction`) Escalonamento (`static`, `dynamic`, `guided`, `runtime`, `auto`)
3	Desafios de programação Seções (`sections`) Sincronização (`nowait`, `critical`, `atomic`, `barrier`, `flush`)
4	Maratona de programação paralela

a execução do *Mooshak*. Verificou-se as máquinas do laboratório no qual foi executado os experimentos para garantir a implementação local dos algoritmos e submissão remota, por meio do sistema em questão.

Os prêmios a serem atribuídos as equipes melhor colocadas foram adquiridos em ocasião anterior à realização da aula.

Operação

No primeiro dia de curso foi solicitado que os alunos constituíssem grupos de até 3 integrantes, os quais foram cadastrados na plataforma. Em um segundo momento os estudantes foram providos de um documento impresso (cuja versão *online* também foi disponibilizada) contendo a descrição e implementação sequencial dos desafios propostos para a aula, além de um resumo da teoria relacionada.

Em cada aula, os pesquisadores conduziram uma apresentação rápida do conteúdo presente no caderno de desafio e, então, utilizou-se o projetor para exibir as pontuações das equipes, enquanto os docentes atuaram para responder às dúvidas dos alunos e verificar o sistema *Mooshak*.

Ao fim de cada aula, os códigos submetidos ainda foram avaliados manualmente e as equipes vencedoras da maratona da aula foram anunciadas e receberam prêmios (bombons). Na última aula, que foi integralmente composta por uma maratona de programação, os vencedores receberam, ao fim da competição, baralhos de truco e bombons.

5.5.4 Resultados

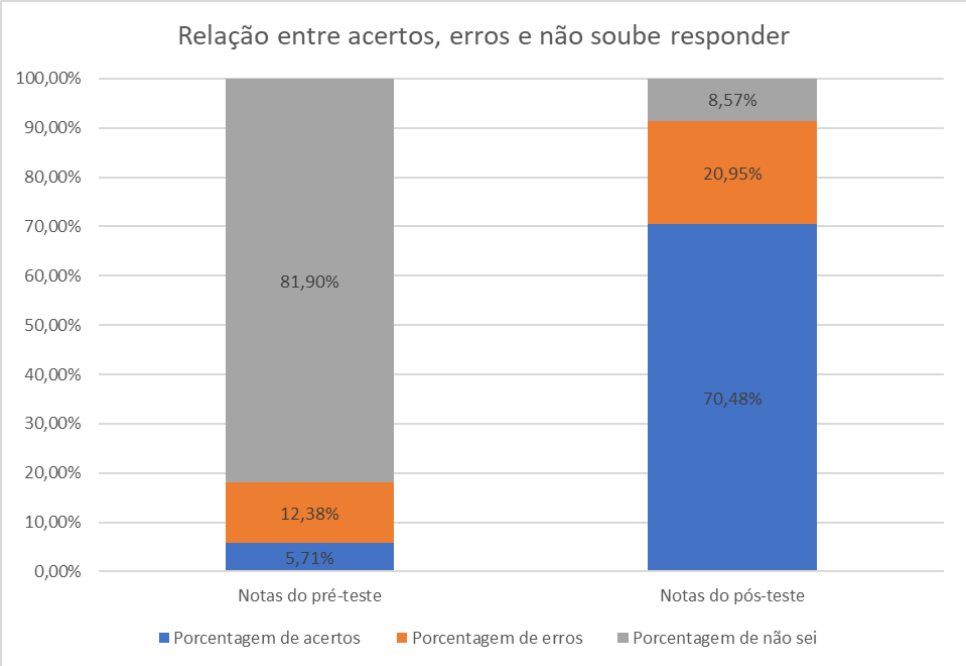
Aprendizado técnico

Foi aplicado um pré-teste, ao início da primeira aula, e um pós-teste no início da última

aula. Cada teste era composto de 10 questões com 5 alternativas cada, em que a última alternativa (e) era "não sei". Foi recomendado que os alunos, caso não soubessem a resposta correta, evitassem arriscar uma alternativa e selecionassem a alternativa *e*.

Um total de 21 alunos responderam a ambos os testes. A Figura 39 ilustra a porcentagem de respostas corretas, incorretas e "não sei". Verifica-se que a maior parte dos estudantes não sabiam ou erraram as respostas do pré-teste, com um percentual de acerto de apenas 5,7%. Já no pós-teste, 70,4% das respostas foram corretas, representando uma evolução considerável do aprendizado entre os duas ocasiões.

Figura 39 – Percentagem de acertos, erros e "não sei"dos pré e pós-testes



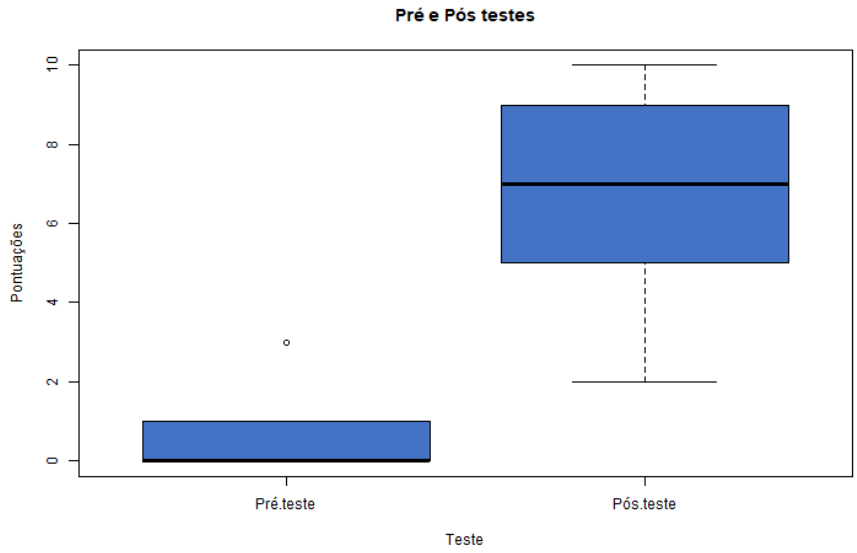
A Figura 40 ilustra a diferença entre as pontuações obtidas pela turma entre o pré-teste e pós-teste. É possível perceber que a evolução do desempenho dos alunos no pós-teste foi significativa, quando comparada com as pontuações obtidas no pré-teste.

No intuito de fundamentar estatisticamente as afirmações acerca do desempenho dos alunos, fez-se algumas análises. Inicialmente, fez-se o teste de *Shapiro-Wilk* para atestar a normalidade dos dados, com nível de significância de 95%, obtendo-se os seguintes valores:

- Pré - teste: W = 0,6853, $p\text{-}valor = 1{,}819 \cdot 10^{-5}$. $p\text{-}valor < 0{,}05$. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0.9241, $p\text{-}valor = 0.1048$. $p\text{-}valor > 0{,}05$. Conclusão: dados seguem a distribuição normal.

Posteriormente, em função da não normalidade do conjunto de dados, fez-se o teste de *Wilcoxon* para amostras pareadas, também com 95% de significância, no intuito de avaliar a

Figura 40 – Comparação entre as pontuações do pré-teste e pós-tese



igualdade (*h0*) ou não (*h1*) das médias obtidas em cada teste, produzindo os seguintes resultados:

- *p-valor* = $6{,}091 \cdot 10^{-5}$, *p-valor* < 0,05. Conclusão: Rejeita-se *h0*, significando que os alunos obtiveram melhores notas, em média, no pós-teste.

Um total de 214 submissões eram esperadas, calculado a partir do produto entre quantidade de equipes presentes e de desafios por aula. Verificaram-se 158 (74%) submissões realizadas e 56 (26%) não realizadas, demonstrando que a maior parte dos alunos se empenhou para resolver os desafios propostos.

Além disso, foi feita a avaliação das submissões feitas em função da sua corretude e qualidade. A média da turma, em critérios de qualidade de código submetido, foi de 85%. A Figura 41 ilustra a qualidade média das submissões realizadas por equipe.

É possível perceber que, em geral, as equipes desenvolveram códigos funcionais e com qualidade e apenas 4 equipes apresentaram média inferior a 85%.

É possível perceber, analisando a Figura 42, que, em média, os alunos obtiveram bom desempenho nos desafios propostos, os quais em apenas 5 desafios a média foi inferior a 85%. É válido notar que as pontuações obtidas também foram proporcionais ao nível de complexidade dos desafios propostos, de modo que nos desafios finais, e por consequência mais difíceis, a qualidade do código desenvolvido, em média, foi mais baixa.

Avaliação qualitativa

No intuito de estabelecer a receptividade dos participantes do presente experimento em relação ao conteúdo, metodologia e materiais empregados, utilizou-se um questionário adaptado do modelo *ARCS* e a proposta de Pesquisa de Interesse de Curso definida em Keller (2010).

Figura 41 – Médias de qualidade de código por equipe

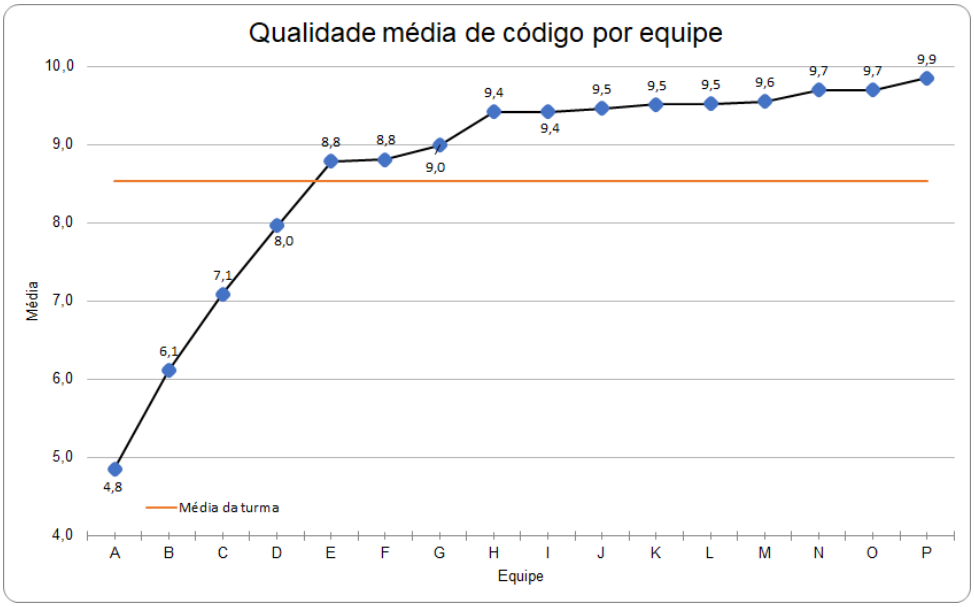
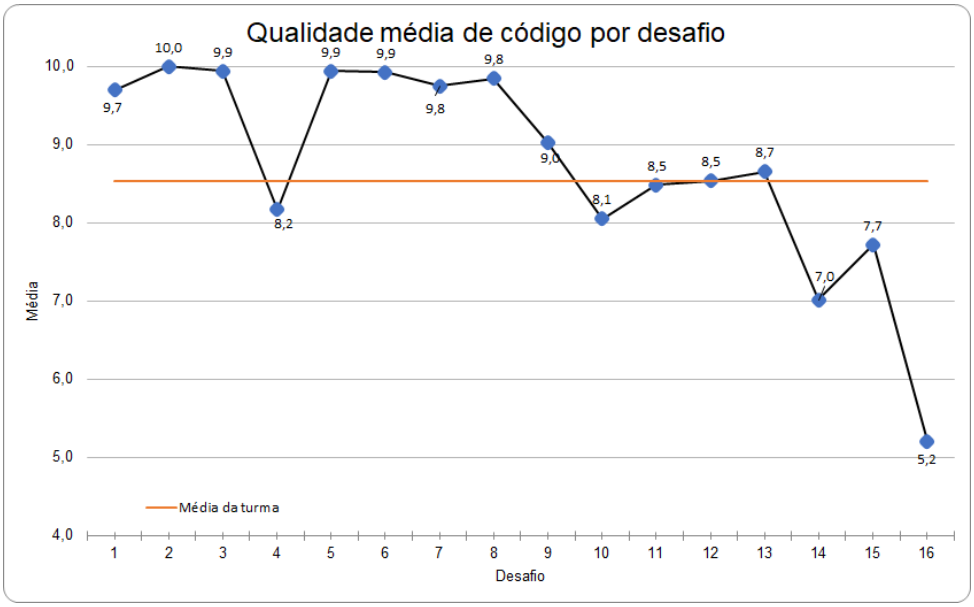


Figura 42 – Qualidade de código média por desafio



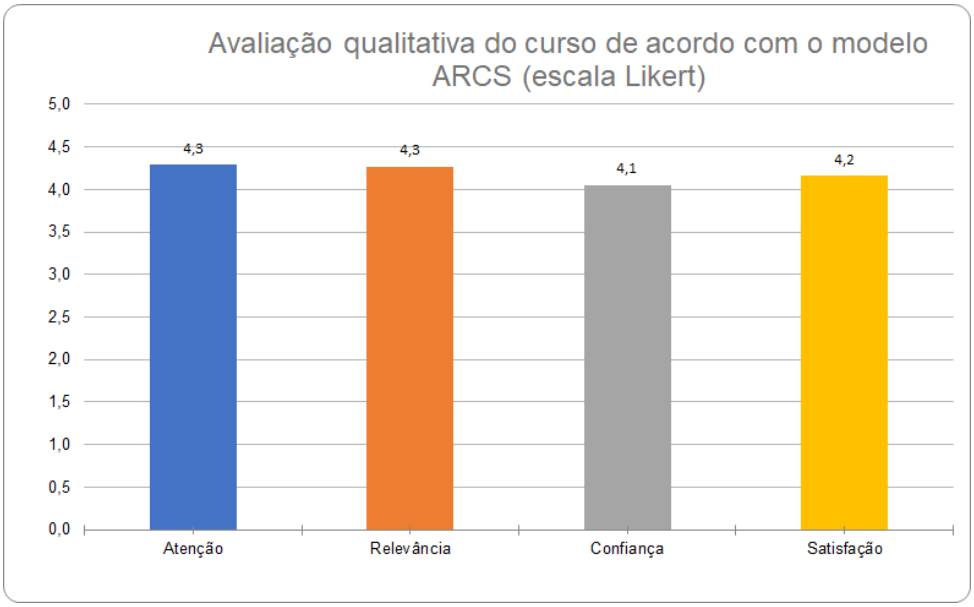
O questionário elaborado pode ser visto em <http://bit.do/qualitativo_openmp_2>.

Um total de 19 alunos responderam ao questionário. Os gráficos que ilustram as respostas para as 34 questões da avaliação qualitativa podem ser acessados em <http://bit.do/graficos_qualitativo_openmp>.

De forma a sumarizar as informações obtidas, elaborou-se um gráfico padronizando as avaliações na escala *likert*. O referido gráfico pode ser visto por meio da Figura 43. É possível perceber, por meio da análise do mesmo, que a avaliação do curso foi bastante positiva, traduzida por uma média de satisfação de 83%.

Além disso, permitiu-se que os participantes do curso fornecessem sugestões, críticas

Figura 43 – Avaliação qualitativa do curso na escala likert



e observações sobre o curso, de maneira livre e anônima. Abaixo, seguem alguns comentários selecionados.

- “Curso muito bom, o conteúdo pode ser bem explicado e de maneira dinâmica e didática aos alunos. Quanto as atividades, elas cobravam realmente o conteúdo apresentado durante as aulas e tinham um grau de dificuldade pertinente ao assunto”.

- “Achei o curso bem organizado. A forma gradual de se introduzir os novos conhecimentos não me deixou perdido em nenhum momento. Entretanto nas duas primeiras aulas não consegui executar um dos exercícios por falta de tempo”.

- “(...) Inicialmente a ideia de uma competição não me pareceu interessante (por eu não gostar muito), e ficou um pouco confuso se era em grupo e etc, com pessoas desconhecidas. Mas ao longo do curso, a questão da competição foi bem legal. Mais por ter um feedback do meu desempenho. Acho que o prêmio ser algo simbólico ajudou nessa questão, diminuindo a ansiedade e competitividade - o que é algo bom e saudável. No conjunto, talvez essa experiência me dê incentivo e coragem pra participar de outras competições(...)”

- “Curso muito bom. Exemplos cotidianos que fazem com que o conteúdo seja aprendido de uma forma melhor. Material muito bom, contando com a plataforma e exemplos já encaminhados. Espero outros cursos como esse no futuro”.

- “A terceira aula teve muito conteúdo, o curso poderia ter mais uma aula pros conteúdos não ficarem tão apertados”.

- “Acredito que na avaliação dos códigos, ao invés de apenas dar uma nota, retornarem de alguma forma o que poderia estar melhor naquele código. Dessa forma, saberemos utilizar de maneira mais eficiente os recursos oferecidos pelo OpenMP”.

5.5.5 Análise

Apresentamos aqui a análise em função dos resultados obtidos, almejando responder as questões de pesquisa estabelecidas na Subseção 5.5.2.

[QP1] Os alunos foram capazes de absorver o conteúdo teórico de *OpenMP*

É possível notar, por meio da análise da Figura 40 e da análise estatística correspondente, que houve uma melhoria significativa entre as pontuações obtidas no pós-teste, em relação ao pré-teste.

Houve um percentual de acertos de 70,4% no pós-teste e de apenas 5,7% no pré-teste. Além disso, a percentagem de ocasiões nas quais os alunos não souberam responderam reduziu de 81,9%, no pré-teste, para 8,5%, no pós-teste.

Portanto, é possível afirmar que houve aprendizado considerável do conteúdo teórico previsto para o experimento.

[QP2] Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP* com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho

Verificou-se uma média de 85% na pontuação atribuída para qualidade do código-fonte submetido na turma. Além disso, pode-se perceber que as equipes desenvolveram códigos com qualidade, ilustrado na Figura 41, e, na maioria dos desafios, a média de qualidade de código foi igual ou superior a 85%, visto na Figura 42. Adicionalmente, 74% dos desafios propostos foram desenvolvidos pelas equipes participantes, representando o interesse dos participantes no conteúdo e implementações propostas.

Dessa forma, é possível afirmar que os alunos foram capazes de desenvolver códigos paralelos corretos e com qualidade considerável.

[QP3] Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos

Com base nas respostas para o questionário qualitativo, foi possível perceber que, em geral, houve um nível de satisfação significativo dos estudantes que participaram do experimento.

A Figura 43 mostra que para todas as categorias consideradas na análise feita, houve um excelente grau de receptividade e, considerando a escala *likert*, houve uma aprovação média de 83% do curso pelos alunos.

Além disso, com base nos comentários providos pelos estudantes, foi possível notar que,

em grande parte, os mesmos ficaram bastante motivados e satisfeitos com o curso, de uma forma geral. Houve, no entanto, críticas sobre a divisão do tempo para a realização das atividades e a quantidade de conteúdo ministrado.

Finalmente, é possível estabelecer que os alunos se sentiram confortáveis e motivados com a metodologia, recursos educacionais, didática e demais ferramentas pedagógicas empregadas no contexto deste experimento.

5.6 Experimento IV: Disciplina de Computação de Alto Desempenho para Engenharia de Computação baseada em *Problem Based Learning*

5.6.1 Escopo

Objetivo

O objetivo deste experimento foi averiguar a capacidade de graduandos em Engenharia da Computação (em estágio intermediário do curso) de aprender o conteúdo de Computação de Alto Desempenho e desenvolver aplicações paralelas com qualidade.

Objeto

O objeto de estudo deste experimento é a aplicação de desafios de programação visando o ensino de conceitos de computação de alto desempenho com base na metodologia *Problem Based Learning* (PBL).

Foco qualitativo

O experimento visou mensurar o desempenho técnico dos alunos quando submetidos à metodologia ativa e, além disso, analisou a evolução dos discentes ao fim da disciplina em critérios de conhecimento teórico e prático de paralelismo.

Perspectiva

Por tratar-se de um estudo experimental, tentou-se conduzir esta avaliação em um ambiente real do objeto sob investigação. Portanto, considerando a investigação de uma abordagem de ensino ativa em sala de aula, o ambiente ideal para realização seria o meio acadêmico.

Contexto

Mediante a perspectiva, optou-se por realizar o experimento na execução da disciplina de Computação de Alto desempenho. O curso foi estruturado como disciplina obrigatória para alunos do 7º período da graduação em Engenharia da Computação na instituição ICMC - USP. Houve um total de 8 alunos matriculados. O curso teve carga horária total de 90 horas. A metodologia de ensino/aprendizado aplicada neste curso foi *Problem Based Learning* (PBL), de

modo que cada aula foi baseada em um ou mais desafios de programação.

5.6.2 Planejamento

Definição das questões de pesquisa

Em razão do contexto deste experimento e do objetivo proposto, podemos elencar as seguintes hipóteses:

QP1 Os alunos foram capazes de aprender o conteúdo teórico de *OpenMP*, *MPI* e *CUDA* (análise dos pré e pós testes para cada modelo de programação)?

QP2 Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP*, *MPI* e *CUDA* corretas e com qualidade? e

QP3 Os alunos se sentiram motivados a aprender os conceitos e desenvolver os desafios propostos?

Seleção de sujeitos

A amostra deste experimento foi selecionada em função dos alunos que atendiam os pré-requisitos definidos pela instituição ICMC USP e foram autorizados a se matricular.

A grade do curso de Engenharia da Computação pode ser vista no *link* <http://bit.do/grade_engComp_2019>, visando contextualizar a situação desses alunos em função das disciplinas já cursadas e o período em que os mesmos se encontravam.

Descrição da Instrumentação

Cadernos de desafios foram fornecidos aos alunos antes de cada aula. Este caderno continha a descrição do desafio e a teoria necessária para a resolução do mesmo. Todos os cadernos produzidos para este experimento, juntamente com os desafios de programação propostos, em versões sequencial e paralela, podem ser obtidas no endereço <http://bit.do/cadernos_de_desafios>.

Em todas as aulas foi utilizado um sistema de apoio a realização de maratonas de programação *Mooshak 2* (descrito na Seção 2.7.1 e disponível em <https://mooshak2.dcc.fc.up.pt/>).

Planejamento de curso

O Quadro 17 relaciona os tópicos ministrados e conceitos associados.

Quadro 17 – Planejamento de curso. Linhas marcadas em azul constituem aulas consideradas para a consolidação deste experimento.

Tópico	Conteúdo
1	Apresentação da disciplina e Introdução à computação paralela: motivação, objetivos, evolução, diferenças entre soluções sequenciais e paralelas
2	Introdução à computação paralela: conceitos básicos
3	Arquiteturas paralelas: introdução, Flynn e memória compartilhada (MIMD)
4	Arquiteturas paralelas: memória distribuída (MIMD)
5	Arquiteturas paralelas: SIMD e GPUs
6	Redes de interconexão
7	Itens que afetam o desempenho; *Speedup* e Eficiência; Amdahl
8	Itens que afetam o desempenho; *Speedup* e Eficiência; Amdahl
9	Projeto de algoritmos paralelos
10	Projeto de algoritmos paralelos
11	Projeto de algoritmos paralelos
12	Projeto de algoritmos paralelos
13	**1a. avaliação da disciplina**
14	Projeto de algoritmos paralelos
15	Programação *multicore*: *OpenMP*
16	Programação *multicore*: *OpenMP*
17	Programação *multicore*: *OpenMP*
18	Programação *multicore*: *OpenMP*
19	Programação em *clusters*: *MPI* Introdução e exercício (funções básicas `send` e `recv`)
20	Programação em *clusters*: *MPI* Exercício com comunicação bloqueante ponto-a-ponto
21	Programação em *clusters*: *MPI* Exercício com comunicação ponto-a-ponto modos diferentes
22	Programação em *clusters*: *MPI* Operações coletivas, incluindo `spawn`
23	**2a. avaliação da disciplina**
24	Programação em *clusters*: *MPI* com *OpenMP*
25	Programação em processadores heterogêneos: Introdução à *CUDA*, GPUs, organização da memória, sincronização e modelos de programação.
26	Programação em processadores heterogêneos: *CUDA* Organização e uso de memória, sincronizção (com exemplos)
27	Programação em processadores heterogêneos: *CUDA* Execução assíncrona
28	Programação em processadores heterogêneos: *CUDA* Desafio: Somatório de produtos escalares
29	Programação em processadores heterogêneos: *CUDA* Desafio: Medidas estatísticas de uma matriz
30	**3a. avaliação da disciplina**

5.6.3 Execução

Preparação

No início do semestre letivo, fez-se o planejamento do curso em função do conteúdo a ser ministrado, categorizando-o por modelo de programação *(OpenMP), (MPI)* e *(CUDA)*. Para cada modelo, elaborou-se o pré-teste e pós-teste e selecionou-se os desafios de programação a serem aplicados.

Em função destes desafios de programação definidos, o conteúdo dos cadernos de desafio foi estruturado e redigido, contendo a descrição dos desafios, suas implementações sequenciais, além de síntese da teoria e referências bibliográficas pertinentes. Antes de cada aula, o material correspondente era liberado, por meio do sistema *Moodle*, em versão digital. Além disso, na ocasião da aula, eram impressos e entregues para cada aluno um exemplar correspondente ao conteúdo da aula em questão.

Adicionalmente, o sistema *Mooshak* foi instalado, configurado e implantado na infraestrutura do *Google cloud*. Foi disponibilizado um endereço IP externo para os alunos acessarem a interface (*webservice*) do sistema de maratona. A cada aula a descrição dos desafios e casos de teste correspondentes eram inseridos no *Mooshak*.

Operação

No primeiro dia de aula, foi solicitado que os alunos se organizassem em duplas, que deveriam ser respeitadas ao longo do semestre letivo. As duplas foram cadastradas no *Mooshak*, de modo que cada aluno recebeu um *id* e senha para acesso. Além disso, o sistema foi demonstrado e foi explicitado como seria feita a avaliação e atribuição de pontuação.

Antes de cada aula, como foi dito, o caderno de desafio era divulgado via *Moodle* e no início da aula sua versão impressa era entregue. Durante a aula, o conteúdo era abordado na sequência prevista no documento, ainda que de maneira mais aprofundada. Os últimos 60 minutos (de um total de 150 minutos) era destinado a implementação dos desafios. Nesta etapa, o professor e aluno auxiliar procuravam atender às dúvidas dos alunos.

A pontuação dos desafios de cada aula era atualizada regularmente, após a avaliação manual dos códigos submetidos pelos pesquisadores e os *scores* e classificação derivada disponibilizada por meio do *Moodle*.

É importante destacar que, por restrições financeiras em função do custo operacional do *Google cloud*, o sistema *Mooshak* e a máquina virtual que o hospedava eram ligados ao início de cada aula e desligados após o fim da mesma, salvo ocasiões nas quais o prazo de submissão dos desafios era estendido.

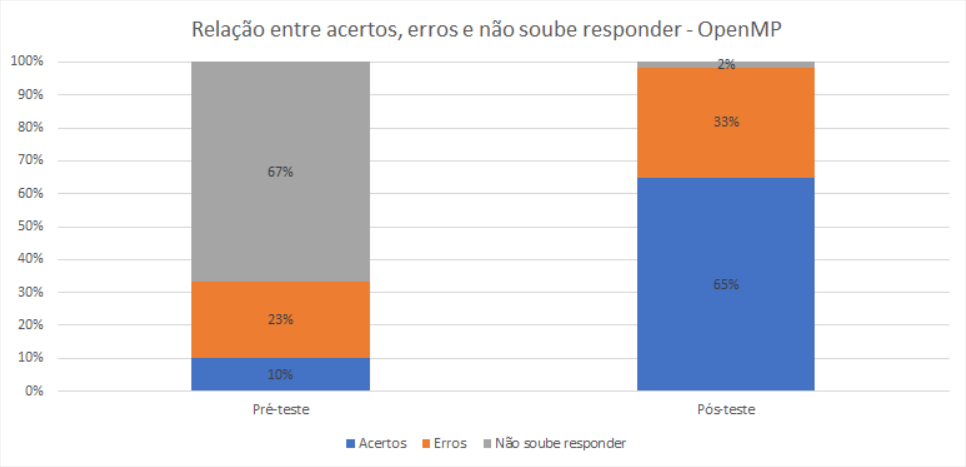
5.6.4 Resultados

Aprendizado técnico

Um primeiro resultado para este experimento trata-se do conjunto das pontuações obtidas a partir da aplicação de um pré-teste e um pós-teste para cada modelo de programação.

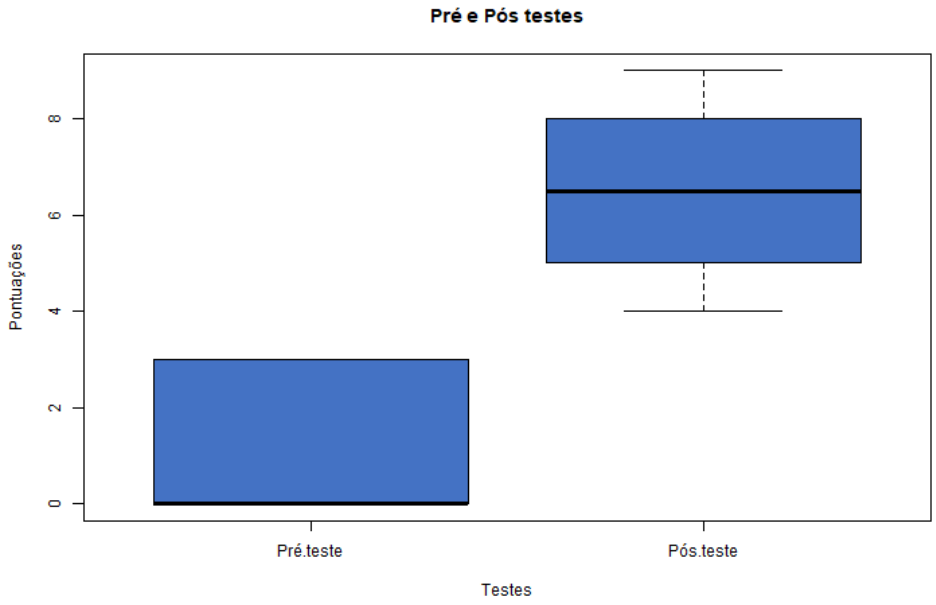
Desta forma, inicialmente foi aplicado um pré-teste, no contexto da primeira aula referente ao módulo de *OpenMP*, e um pós-teste , na última aula deste módulo. Um total de 6 alunos realizam ambos os testes. A Figura 44 exibe o percentual de respostas corretas, incorretas e "não sei". É possível notar que a maior parte dos estudantes não sabiam ou responderam de forma incorreta as questões do pré-teste, no qual houve apenas 10% de acertos. Já no pós-teste, verificou-se uma percentagem de 65% de acertos, indicando uma melhoria significativa do desempenho dos alunos.

Figura 44 – Percentagem de acertos, erros e "não sei"dos pré e pós-testes - OpenMP



A Figura 45 representa a comparação entre as pontuações das pontuações obtidas pelos alunos no pré-teste e pós-teste.

Figura 45 – Comparação entre as pontuações do pré-teste e pós-tese - OpenMP



No intuito de corroborar os indícios apontados pelas figuras apresentadas, fizeram-se análise estatística sobre os dados obtidos.

Realizou-se o teste de *Shapiro-Wilk* para verificar a normalidade dos dados, com nível de significância de 95%. Os seguintes valores foram obtidos:

- Pré - teste: W = 0,6398, *p-valor* = 0,0013. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0,9818, *p-valor* = 0,9606. *p-valor* > 0,05. Conclusão: dados seguem a distribuição normal.

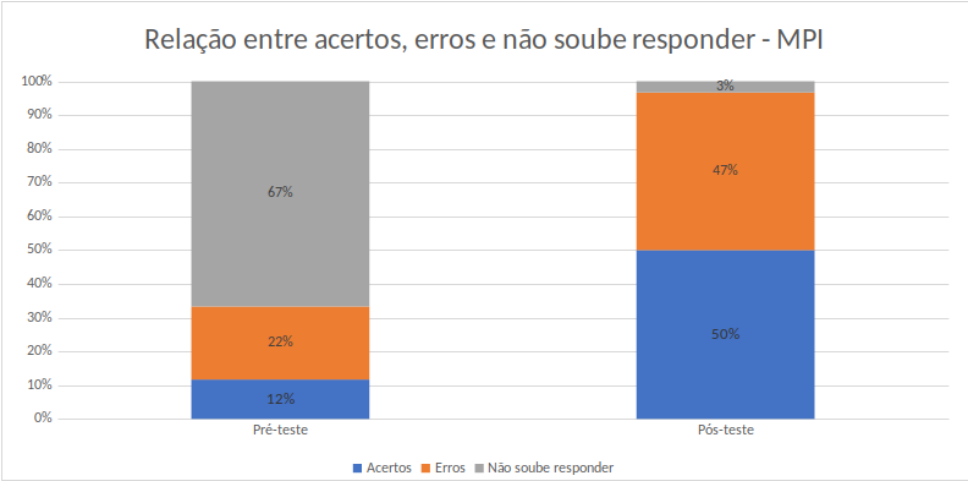
O teste de *Wilcoxon* para amostras pareadas, também com 95% de significância, foi utilizado para avaliar a igualdade (*h0*) ou não (*h1*) das médias obtidas nos testes, resultando no seguinte:

- *p-valor* = 0,0360, *p-valor* < 0,05. Conclusão: Rejeita-se *h0*, significando que os alunos obtiveram melhores notas, em média, no pós-teste.

É possível perceber, portanto, que houve aprendizado para o conteúdo teórico de OpenMP.

Posteriormente, foi feito um pré-teste e pós-teste para o conteúdo de MPI. A Figura 46 apresenta o percentual de acertos, erros e "não sei". Percebe-se, por meio da referida figura, que o percentual de acertos no teste evoluiu de 12% para 50%, significando uma melhoria razoável no desempenho dos alunos.

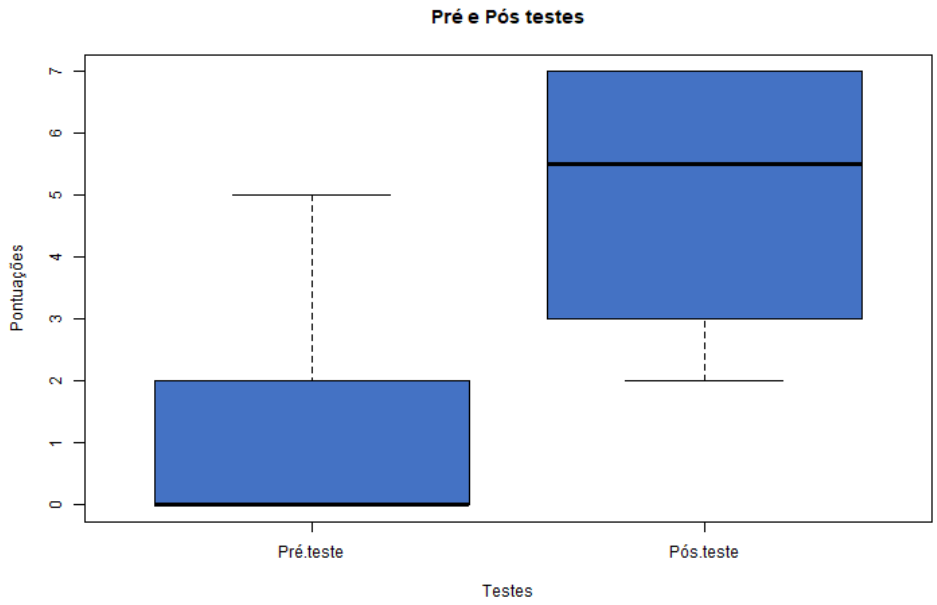
Figura 46 – Percentagem de acertos, erros e "não sei"dos pré e pós-testes - MPI



A Figura 47 mostra a comparação entre as pontuações obtidas pela turma para o conteúdo de *MPI*.

O mesmo procedimento estatístico foi replicado para a análise do pré-teste e pós teste do conteúdo de *MPI*. Primeiramente, aplicou-se o teste de *Shapiro-Wilk*, com 95% de significância, com o seguinte resultado:

Figura 47 – Comparação entre as pontuações do pré-teste e pós-tese - MPI



- Pré - teste: W = 0,6844, *p-valor* = 0,0042. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0,8904, *p-valor* = 0,3204. *p-valor* > 0,05. Conclusão: dados seguem a distribuição normal.

Aplica-se, então, o teste de *Wilcoxon* para definir a equivalência (*h0*) ou não (*h1*) das médias de cada teste, resultando no seguinte:

- *p-valor* = 0,0731, *p-valor* > 0,05. Conclusão: Não rejeita-se *h0*, significando que ambas médias são estatisticamente equivalentes.

Ainda que o resultado do teste mostre que as médias do pré-teste e pós-teste são equivalentes, é possível perceber, com base nas figuras 46 e 47, que houve 38% mais acertos no pós-teste do que no pré-teste. Além disso, é importante considerar a alta complexidade deste conteúdo e o tamanho reduzido da amostra, que implica em alta variabilidade dos dados, para interpretar estes resultados.

Para o conteúdo de CUDA, também aplicou-se um pré-teste e pós-teste, de maneira similar aos outros modelos de programação. Neste caso, a Figura 48 mostra a percentagem de acertos, erros e "não sei"da turma. Percebe-se que houve uma melhora significativa no desempenho da turma, ilustrada por uma taxa de acertos de 14% no pré-teste e de 73% no pós-teste.

De forma complementar, a Figura 49 contém a comparação entre as pontuações obtidas em ambos testes.

Figura 48 – Percentagem de acertos, erros e "não sei"dos pré e pós-testes - CUDA

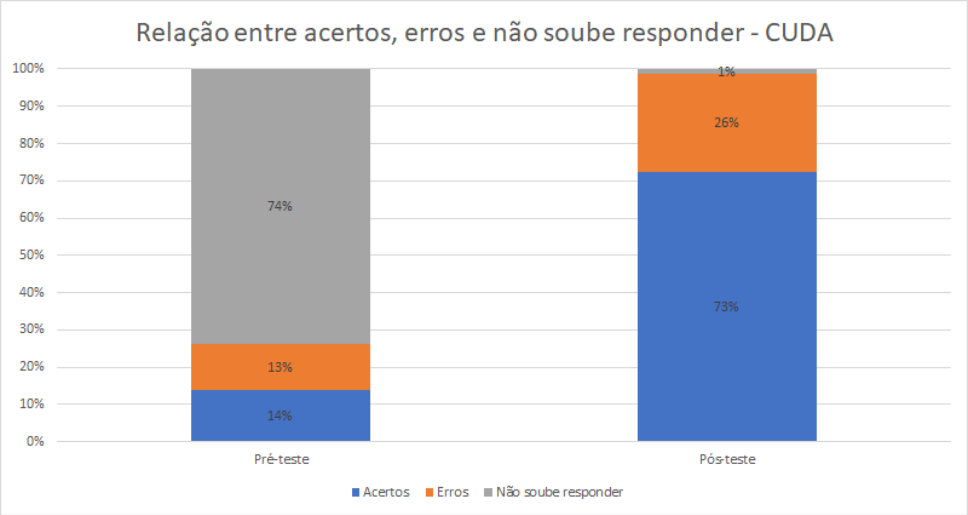
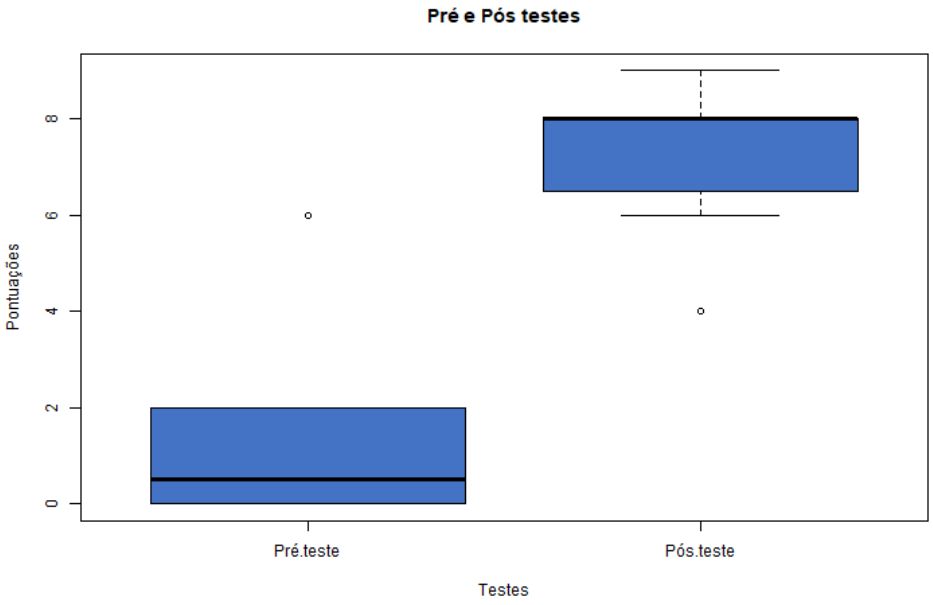


Figura 49 – Comparação entre as pontuações do pré-teste e pós-tese - CUDA



Novamente, realiza-se o teste de *Shapiro-Wilk* com 95% de confiança produzindo o seguinte:

- Pré - teste: W = 0,7299, *p-valor* = 0,0048. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0,8395, *p-valor* = 0,0745. *p-valor* > 0,05. Conclusão: dados seguem a distribuição normal.

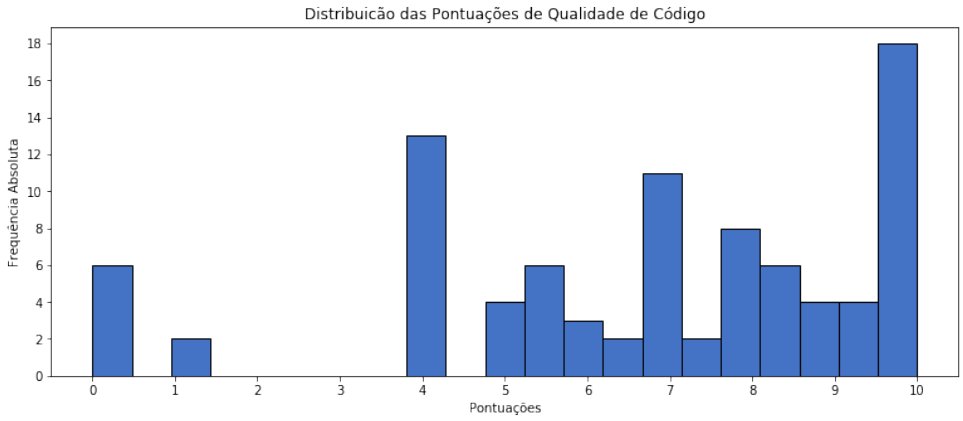
Finalmente, com intuito de estabelecer se as médias obtidas para cada teste são iguais (*h0*) ou não (*h1*), aplica-se o teste de *Wilcoxon*, cujo resultado está disposto a seguir:

- *p-valor* = 0,0222, *p-valor* < 0,05. Conclusão: Rejeita-se *h0*, significando que os alunos obtiveram melhores notas, em média, no pós-teste.

Desta forma, é possível concluir que houve, de fato, um progresso notável do desempenho dos alunos no pós-teste, em relação ao pré-teste.

Um outro resultado trata-se do conjunto das pontuações obtidas pelos alunos em critérios de qualidade de código. A Figura 50 apresenta a distribuição das notas em formato de histograma. É possível perceber, por meio da análise desta figura, que houve considerável variabilidade dos dados, embora a maior frequência de *scores* apresentados está contido entre 9,5 e 10 pontos, o que indica que a turma desenvolveu uma quantidade relevante de aplicações paralelas de qualidade.

Figura 50 – Histograma das pontuações para qualidade de código desenvolvido



A qualidade média de código da turma foi de 6,6 pontos. Na Figura 51 pode-se verificar as médias obtidas por equipe. Nota-se que, em geral, as equipes obtiveram bom desempenho nas atividades realizadas, na quais apenas um time obteve aproveitamento inferior a 60% da nota de qualidade de código.

A Figura 52 apresenta uma visualização da qualidade média de código por desafio de programação. É possível verificar que houve alta variabilidade nas pontuações obtidas, ainda que perceba-se na maioria dos desafios (58%) a nota foi superior a média da turma.

É possível visualizar a qualidade média dos códigos-fonte desenvolvidos na Figura 53. Neste caso, verifica-se que a pontuação, ainda que com alta variabilidade devida principalmente ao tamanho da amostra, foi razoavelmente superior nas atividades referentes ao modelo de programação *OpenMP*, seguido por *CUDA* e finalmente *MPI*, no qual a inferioridade da qualidade dos códigos desenvolvidos reflete parcialmente a dificuldade apresentada pelos alunos ao realizarem o pré-teste e pós-teste deste conteúdo.

Figura 51 – Qualidade de código média por equipe

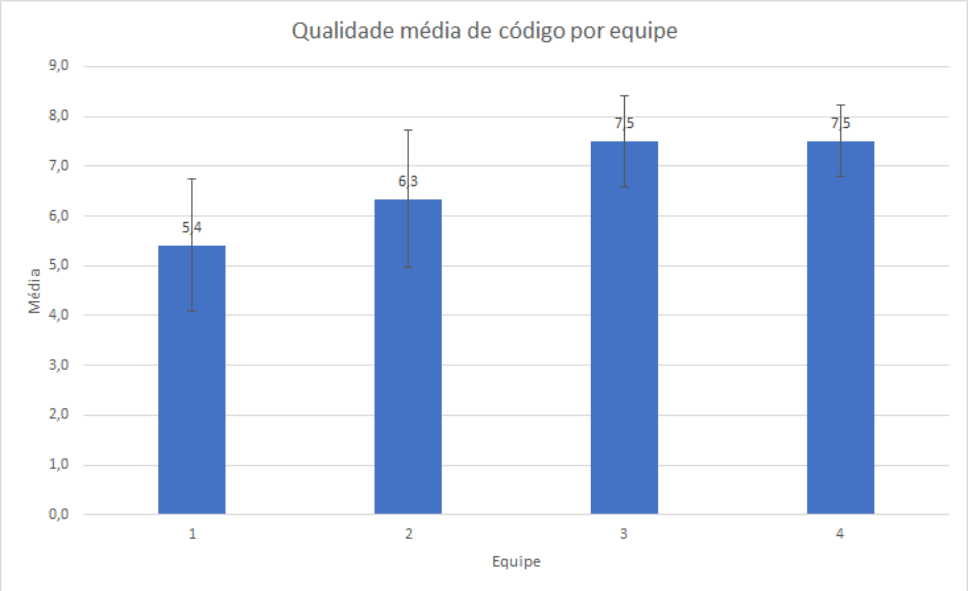
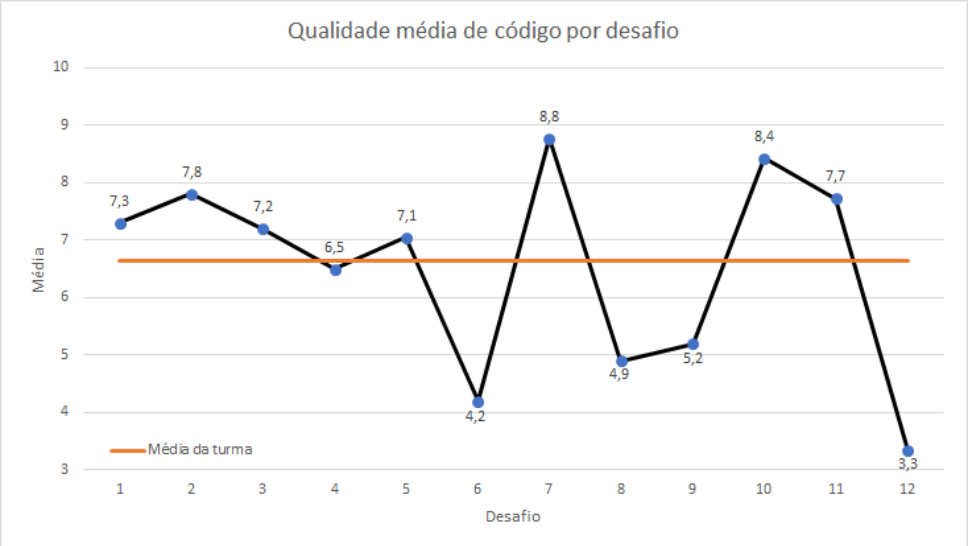


Figura 52 – Qualidade de código média por desafio de programação



Avaliação qualitativa

Foi aplicado, ao fim da disciplina, um questionário adaptado do *Course Interest Survey*, de Keller (2010). As questões desta avaliação podem ser vistas em <http://bit.do/teste_qualitativo_cad>.

Os gráficos contendo as respostas obtidas podem ser consultados no *link* <http://bit.do/graficos_respostas_cad>. O total de alunos da turma (8) respondeu o questionário.

A Figura 54 sintetiza as pontuações obtidas para cada uma das categorias do modelo *ARCS*. É possível perceber, por meio do estudo do referido gráfico, que, em geral, os alunos que participaram do experimento ficaram satisfeitos com o mesmo, ilustrado por meio de uma aprovação média de 74%.

Figura 53 – Qualidade de código média por modelo de programação

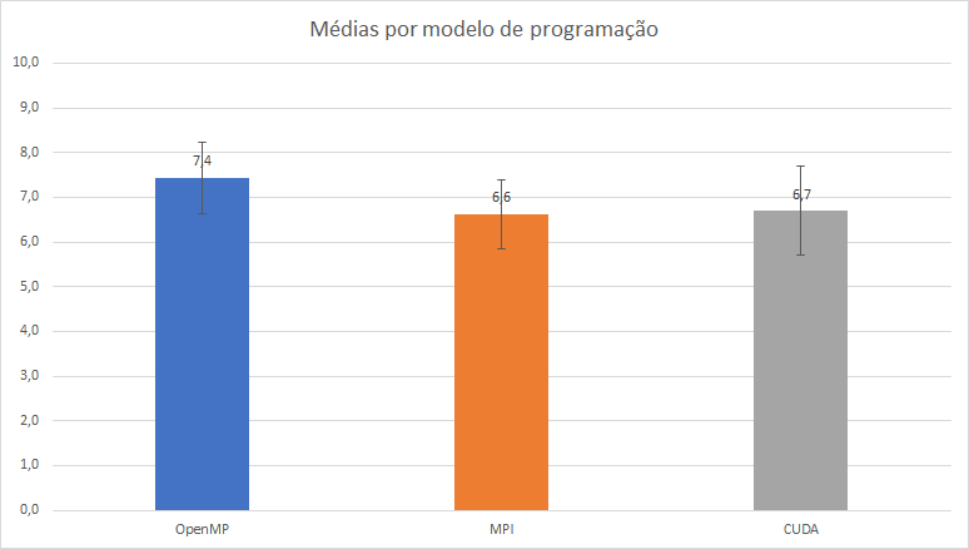
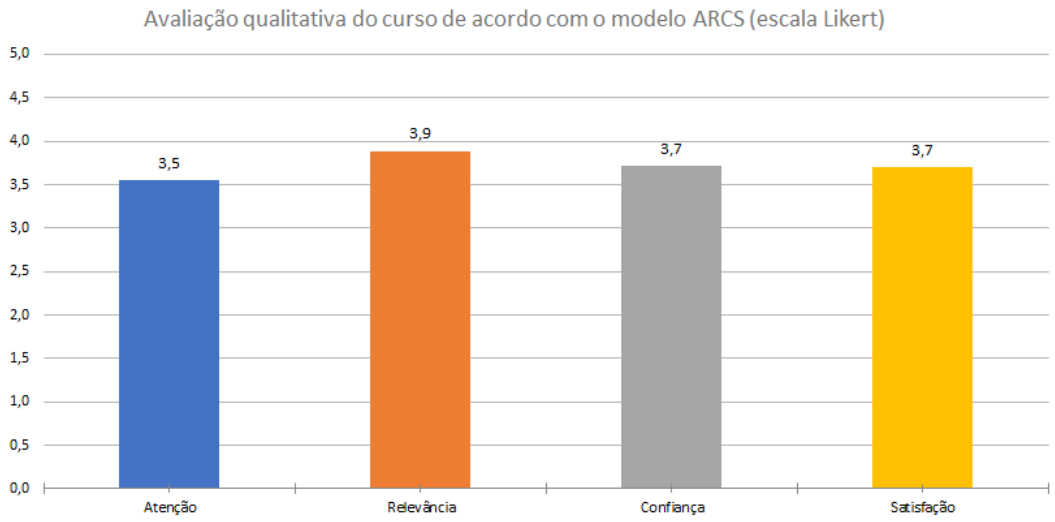


Figura 54 – Avaliação da disciplina na escala likert



Além disso, houveram dois comentários, listados a seguir, que explicitam o ponto de vista dos estudantes que cursaram a disciplina.

- "Foi mais legal do que eu imaginei que seria, pois tenho dificuldade com o tema. Acredito que essa forma prática que o curso foi dado foi boa, porque a gente consegue ver aplicações reais de programação paralela. Acredito que, em uma próxima vez, mais aulas dedicadas apenas a resolver desafios (sem conteúdo em si, apenas prática) beneficiará os alunos".

- "O curso foi bem dado, mas as atividades deveriam ser entregues com antecedência".

5.6.5 Análise

De forma similar aos experimentos anteriormente descritos, utiliza-se esse espaço com objetivo de responder às questões de pesquisa previamente definidas. Desta forma, apresentam-se as avaliações realizadas sobre os resultados obtidos.

[QP1] Os alunos foram capazes de aprender o conteúdo teórico de *OpenMP*, *MPI* e *CUDA*

Foi possível perceber que os alunos foram capazes de assimilar o conteúdo teórico para cada um dos três modelos de programação paralela abordados na disciplina.

Em um primeiro momento, por meio das Figuras 44, 45 e da análise estatística complementar, que houve uma melhoria de 10% para 65% de acertos no pós-teste, em relação ao pré-teste, representando o aprendizado do conteúdo teórico de *OpenMP*.

Para o conteúdo de *MPI*, verificou-se, nas Figuras 46, 47 que houve um aumento de 12% para 50% no aproveitamento teórico de *MPI*. Embora a análise estatística correspondente aos referidos testes tenha apontado a equivalência das médias, salienta-se que o tamanho reduzido da turma e alta variabilidade das notas contribuíram, também, para este resultado.

Em relação ao conteúdo de *CUDA*, foi possível perceber, assim como ilustrado nas Figuras 48 e 49 e corroborado pela análise estatística relacionada, que houve uma evolução razoável na quantidade de acertos (73%) no pós-teste em relação ao pré-teste (14%). Desta forma, é possível afirmar que houve aprendizado referente ao conteúdo teórico de *CUDA*.

[QP2] Os alunos foram capazes de desenvolver aplicações paralelas em *OpenMP*, *MPI* e *CUDA* corretas e com qualidade, considerando bom uso dos recursos do modelo de programação, paralelismo e desempenho.

Os resultados obtidos para a qualidade dos códigos-fontes desenvolvidos pelos alunos participantes deste experimentos mostram uma média de 66% da turma para esta métrica.

Foi visto que, em geral, as equipes foram capazes de desenvolver códigos funcionais e com qualidade, de modo que que apenas uma equipe conquistou pontuação média neste quesito inferior à 60% da nota. Esta informação pode ser verificada na Figura 51.

Além disso, sob uma perspectiva dos desafios de programação propostos, em 58% das atividades, a nota média apresentada pelas equipes foi superior à média da turma.

Assim, é possível dizer que os alunos foram, de fato, capazes de produzir códigos com corretude e qualidade satisfatórias, em função dos critérios estabelecidos.

[QP3] Os alunos se sentiram confortáveis e motivados a aprender os conceitos propostos?

Com base na aplicação do questionário baseado no instrumento *CIS* e no modelo *ARCS*, verifica-se uma satisfação geral da turma de 74%.

Além disso, a Figura 54 ilustra, na escala *likert*, as médias obtidas para cada uma das categorias do *ARCS*. É possível notar que, para todos as categorias avaliadas a pontuação média foi de pelo menos 70%.

Finalmente, analisando os comentários providos pelos alunos, nota-se a aprovação com a forma em que foi conduzido este experimento e sua qualidade geral. Entretanto, houve queixas sobre o tempo para realização das atividades e da divulgação das tarefas.

5.7 Análises dos resultados

Em função dos resultados apresentados nas Seções anteriores, deseja-se validar a proposta e objetivos deste trabalho sintetizado o que foi observado, ao longo dos experimentos conduzidos.

Foram realizados quatro experimentos distintos, com a participação total de 235 alunos, de 15 cursos de graduação ou pós-graduação, os quais apenas 3 cursos (Ciências de Computação, Engenharia da Computação e Sistemas de Informação) eram diretamente relacionados à computação. Além disso, foram utilizadas metodologias de ensino-aprendizado tradicional, *Team Based Learning*, *Problem Based Learning* e dois sistemas de apoio a realização de maratonas de programação, *mooshak* e *PC*^{2}.

Os principais elementos pedagógicos comuns a todos experimentos, os quais constituem no foco da presente dissertação, são os desafios de programação. Foram desenvolvidos e aplicados 67 desafios de programação, que abordaram diferentes conteúdos, situações e apresentaram nível de dificuldade variável.

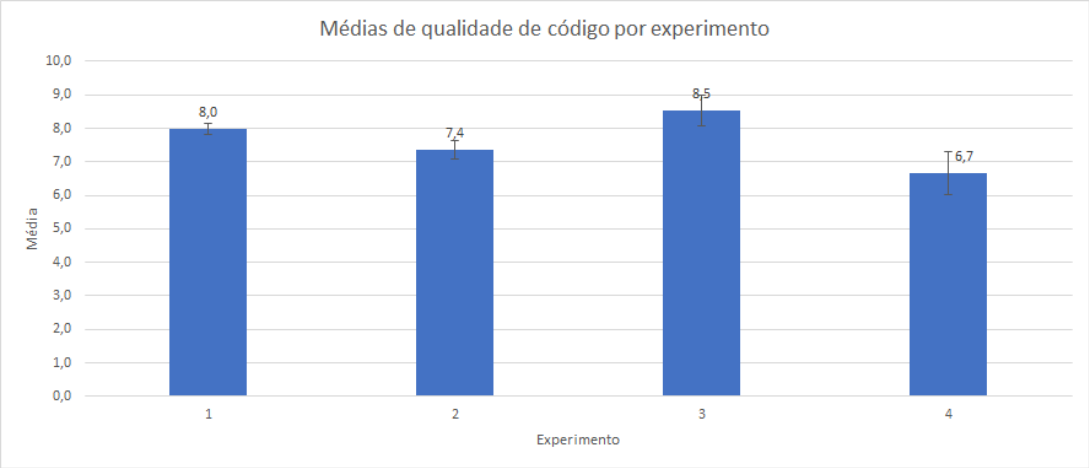
Em vista do objetivo geral deste trabalho e da hipótese estabelecida na Seção 1.5 avaliou-se, como primeira questão de pesquisa, a absorção de conhecimento teórico, por meio de pré-testes, pós-testes e instrumentos equivalentes, obtendo-se, em todos os casos, comprovação do aprendizado dos alunos nesse quesito. Desta forma, é possível afirmar que os alunos foram capazes de aprender e absorver o conteúdo teórico referente aos modelos de programação utilizados e demais conteúdos abordados e, por consequência, que o uso de desafios de programação é um instrumento eficaz como recurso educacional aliado ao ensino-aprendizado de programação paralela.

A segunda questão de pesquisa visa a avaliação da qualidade de código-fonte referente aos desafios de programação. A Figura 55 ilustra as médias de qualidade de código obtidas para cada experimento. Pode-se perceber uma média geral de 76% de aproveitamento na pontuação válida para qualidade de código.

Desta forma, é possível perceber que os desafios de programação solucionados, em média, consistiram de códigos-fonte corretos e com qualidade, considerando bom uso das estruturas, boas práticas e modelos de programação, além de desempenho das soluções desenvolvidas.

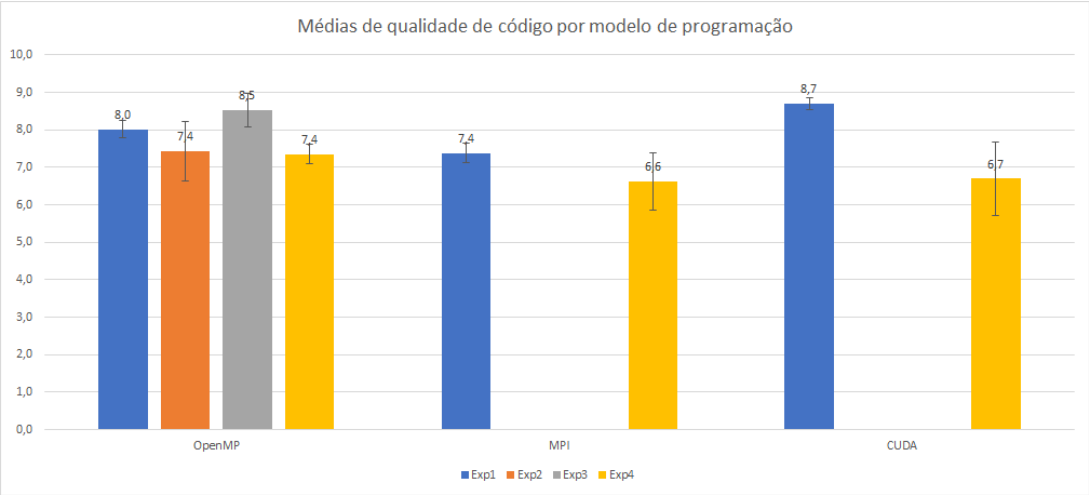
A Figura 56 apresenta as médias obtidas por modelo de programação e experimento.

Figura 55 – Médias de qualidade de código por experimento



Pode-se perceber que os discentes desenvolveram códigos de qualidade para todos os modelos de programação, com destaque para *OpenMP*, com um percentual de 78% de aproveitamento em qualidade de código, seguido por *CUDA*, com 77% e, por fim, *MPI*, com 70%. Neste caso, nota-se que os alunos apresentaram mais dificuldade no modelo de programação *MPI*, o que foi refletido também na pontuação obtida nas avaliações quantitativas.

Figura 56 – Médias de qualidade de código por modelo de programação



A terceira questão de pesquisa visa mensurar a satisfação dos alunos. Foi possível perceber que, em todos os experimentos realizados, os alunos se sentiram satisfeitos e motivados com a metodologia, conteúdo, materiais, recursos educacionais e didática, visto que para todos os experimentos, a aprovação média dos alunos em escala *likert* (independente do uso do questionário *Course Interest Survey*) foi superior a 72%.

Desta forma, com a análise das respostas obtidas para as questões de pesquisa propostas, é possível perceber que os alunos foram capazes de absorver o conteúdo teórico proposto, desenvolver códigos paralelos de qualidade e se sentiram satisfeitos com a forma como os cursos foram conduzidos o que valida a proposta definida inicialmente, na Seção 1.5.

Adicionalmente, faz-se uma análise objetivando verificar se houve diferença estatística entre as pontuações médias de qualidade de código obtidas pelos alunos de cursos de extensão (amostra de 76 estudantes) e das disciplinas obrigatórias (amostra de 159 estudantes).

Com este objetivo, realiza-se o teste de *Shapiro-Wilk* com 95% de confiança, para determinar a normalidade ou não dos dados.

- Pré - teste: W = 0,9595, *p-valor* = 0,0001. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Pós - teste: W = 0,92758, *p-valor* = 0,0012. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.

Assim, devido a não normalidade dos dados, utiliza-se o teste de *Wilcoxon*, para amostras de tamanho variado e não pareadas - *Wilcoxon rank-sum test* (WALPOLE *et al.*, 1993), com intuito de estabelecer se as médias de qualidade de código obtidas são iguais (*h0*) ou não (*h1*).

- *p-valor* = 0,438 *p-valor* > 0,05. Conclusão: Não se rejeita *h0*, significando que as médias de qualidade de código verificadas nos cursos de extensão são estatisticamente equivalentes às pontuações médias apresentadas nas disciplinas obrigatórias.

Desta forma, é razoável afirmar que os códigos desenvolvidos em cursos de extensão foram tão bons quanto aqueles implementados em disciplinas obrigatórias, considerando os critérios adotados para avaliação de qualidade de código.

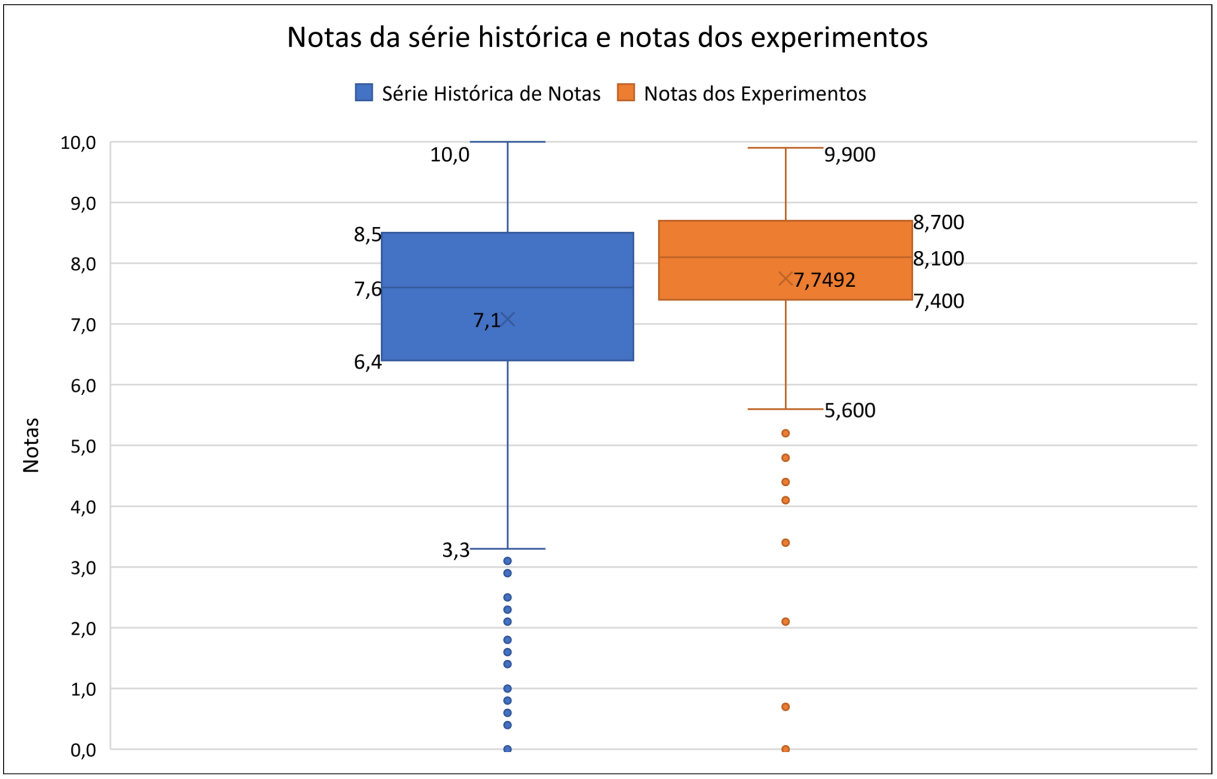
Finalmente, compilaram-se as notas finais obtidas nas disciplinas de programação paralela realizadas entre 2004 e 2017 no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo.

A Figura 57 mostra a dispersão das pontuações das disciplinas anteriores e dos experimentos descritos nessa dissertação. O *boxplot* mostra que o ensino utilizando desafios teve 75% das melhores notas acima de 7,4, enquanto as turmas anteriores tiveram 75% das melhores notas acima de 6,4, apenas. Além disso, a pontuação mais baixa acima dos *outliers* para os experimentos foi de 5,6, enquanto para as disciplinas anteriores foi de 3,3.

Novamente, fez-se o teste de *Shapiro-Wilk* com 95% de confiança, para verificar a normalidade das amostras.

- Disciplinas: W = 0,8706, *p-valor* = $2{,}2.10^{-16}$. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.
- Experimentos: W = 0.76489, *p-valor* = $2{,}2.10^{-16}$. *p-valor* < 0,05. Conclusão: dados não seguem a distribuição normal.

Figura 57 – Comparação das notas das disciplinas anteriores e dos experimentos



Em função da não normalidade dos dados, realizou-se também o *Wilcoxon rank-sum test* comparando as amostras, visando estabelecer se são iguais (*h0*) ou não (*h1*).

- *p-valor* = $9{,}49.10^{-6}$ *p-valor* < 0,05. Conclusão: Rejeita-se *h0*, significando que as pontuações obtidas nas disciplinas não são estatisticamente equivalentes do que àquelas verificadas nos experimentos realizados.

Desta forma, embora a diferença entre suas médias seja pequena (7,1 para as disciplinas anteriores e 7,7 para os experimentos), a análise estatística mostrou que os escores obtidos nos experimentos realizados no contexto da presente dissertação foram maiores do que os das disciplinas anteriores.

CAPÍTULO

6

CONCLUSÕES

6.1 Considerações Iniciais

Este capítulo apresenta as conclusões extraídas a partir dos principais resultados deste trabalho. O capítulo elenca contribuições técnicas e motivacionais para o aprendizado da programação paralela e caracteriza os artefatos gerados na condução das pesquisas realizadas. Por fim, são apresentadas as limitações identificadas, produções científicas e, finalmente, possíveis extensões futuras desta investigação.

6.2 Caracterização da Pesquisa Realizada e seus Resultados

Conforme estabelecido na Seção 1.6, o principal foco deste projeto foi avaliar o uso de desafios de programação e seu impacto para o aprendizado de programação paralela, considerando a absorção de conhecimento teórico, corretude e qualidade das implementações paralelas desenvolvidas, além da motivação dos alunos com o ensino proposto e o quão confortáveis os mesmos se sentiram ao serem expostos a uma abordagem intrinsecamente competitiva.

Com intuito de atingir este objetivo geral, fez-se, inicialmente, uma revisão sobre a fundamentação teórica relacionada ao processo de ensino-aprendizagem, abordagens, métodos e técnicas relacionadas, tais como *Team Based Learning* e *Problem Based Learning*, ambas aplicadas no contexto desta dissertação. Adicionalmente, as principais terminologias e recursos associados a maratonas e desafios de programação foram conceituados, destacando os sistemas PC^2 e *mooshak* que foram utilizados no âmbito deste trabalho.

Em um segundo momento, conduziu-se uma análise sobre os preceitos relacionados ao ensino-aprendizado de computação de alto desempenho tais como definidos pela *ACM* e *IEEE*. Além disso, as principais metodologias e lacunas relacionadas ao ensino e aprendizado de

programação paralela foram elencadas.

Posteriormente, um mapeamento sistemático foi conduzido, visando determinar o estado da arte, no que se refere às metodologias de ensino-aprendizado e recursos educacionais empregados no contexto de programação paralela. Por meio desta análise, foi possível contabilizar e categorizar as publicações encontradas por ano, tipo e recursos pedagógicos empregados; além de identificar tendências e lacunas na linha de pequisa relacionada ao tema.

A partir dessa análise inicial, foi possível delinear os planejamentos de experimentos, hipóteses e questões de pequisa a serem investigadas. Em função da complexidade do processo pedagógico em computação de alto desempenho, resultados relevantes acerca do uso de maratonas e desafios de programação e limitações das abordagens propostas na literatura, julgou-se procedente avaliar a aplicação de desafios de programação como recurso educacional, independente de metodologia ou tecnologia, e centralizador do processo pedagógico.

Desta forma, quatro experimentos foram realizados variando-se a caracterização dos participantes (curso, turma, período), metodologia (tradicional, *Team Based Learning* ou *Problem Based Learning*), uso ou não de sistemas de maratona de programação (PC^2, *Mooshak*) e recursos educacionais (material utilizado no *TBL*, e cadernos de desafio utilizados nos cursos baseados em *PBL*).

A utilização do sistema *Mooshak* foi fundamentada nas publicações de Giménez (2016) e Almeida *et al.* (2012), e o uso do PC^2 foi feito de maneira similar. É importante destacar que as publicações encontradas que detalham o uso de maratonas de programação para o ensino de programação paralela utilizam o *Mooshak*.

Pode-se perceber que o sistema PC^2, ainda que possua uma documentação mais completa do que o sistema *Mooshak*, tem usabilidade inferior e sua configuração é consideravelmente mais complexa do que o último. Além disso, os recursos de personalização do PC^2 são mais limitados do que os oferecidos pelo *Mooshak*. O *Mooshak*, embora ofereça diversos recursos a mais do que PC^2 carece de uma documentação detalhada, o que acaba dificultando a configuração e utilização do *software*.

Notou-se que a utilização de ambos sistemas foi positiva, facilitando a avaliação dos códigos-fontes desenvolvidos pelos estudantes e possibilitando o *feedback* imediato sobre a corretude das implementações desenvolvidas. Entretanto, percebeu-se que os sistemas apresentam limitações derivadas de sua documentação escassa e seu propósito original, visto que ambas ferramentas não foram projetadas com propósito de recursos educacionais.

Os sistemas podem sim ser utilizados para o ensino, ainda que isto demande um esforço considerável do professor, dado que o mesmo precisa configurar, implantar e monitorar as ferramentas. Além disso, notou-se que este tipo de *software* não é mandatório para a aplicação de desafios de programação, ainda que ofereça vantagens para essa atividade. Entretanto, existem funcionalidades que poderiam favorecer o uso desse tipo de sistema no ensino, como

templates para simplificar a configuração dos ambientes, guias orientados ao professor, métricas de avaliação customizáveis (as ferramentas só oferecem análise em função da corretude e o quão rápido um usuário conseguiu submeter a solução correta), tratamento adequado para os códigos paralelos (estes sistemas tratam os modelos de programação paralela como qualquer linguagem de programação sequencial), capacidade para processamento de casos de teste suficientemente grandes (os sistemas não são adequados ao processamento de grandes volumes de dados, o que, em programação paralela, é fundamental para aferir o desempenho de uma solução).

Os resultados desses experimentos foram caracterizados em função do desempenho dos participantes em relação à absorção de conhecimento teórico (avaliação quantitativa), qualidade dos códigos desenvolvidos (corretude, bom uso dos recursos do modelo e linguagem de programação, e desempenho) e motivação dos alunos para estudar o tema (avaliação qualitativa por meio de formulário customizado ou baseada no *Course Interest Survey* (KELLER, 2010).

Considerando uma escala de 0 - 20 % (muito ruim), 20 - 40 % (ruim), 40 - 60 % (regular), 60 - 80 % (muito bom) e 80 - 100 % (ótimo) (STUFFLEBEAM; CORYN, 2014), verificou-se que, para todos os experimentos realizados, a performance dos estudantes participantes foi muito positiva, em todas as métricas consideradas, ilustrada por valores médios de pontuação em avaliações teóricas de 72,6% e máximos de 96%; qualidade de código média de 76% e máxima de 85%; e percentual de motivação médio de 76,5%, com um máximo de 83%.

Em função da hipótese definida na Seção 1.5, os resultados dos experimentos mostraram que é possível aprender programação paralela com o uso de desafios de programação, independentemente do método pedagógico ou recurso educacional utilizado. Os desafios de programação constituem um importante, eficaz e flexível recurso educacional que pode ser aplicado em diferentes contextos, tanto em cursos de extensão quanto em disciplinas regulares de cursos de graduação, com alunos de diferentes períodos ou cursos, desde que adaptados para o público alvo e que exista uma preparação de curso adequada. Conclui-se também que os desafios contribuem para a motivação dos estudantes, visto que um ambiente de competição saudável aliado a premiações, ainda que simbólicas, estimulam a participação dos alunos e o desenvolvimento de soluções criativas.

6.3 Principais Contribuições

Dentre principais contribuições deste trabalho, destacam-se:

- Levantamento bibliográfico contemplando o estado da arte referente ao ensino e aprendizado de programação paralela e disciplina relacionadas, metodologias e recursos educacionais utilizados no processo pedagógico deste tema, e classificações das publicações por cronologia e tipo;
- Desenvolvimento de artefatos constituindo Recursos Educacionais Abertos:

  - Proposta, descrição e implementação de desafios de programação, disponíveis em <http://tiny.cc/desafios_programacao_para>;
  - Compilação de material de apoio para realização de maratonas de programação contendo tutoriais para o PC^2 e *Mooshak 2*, instalação de *CUDA*, criação de máquina virtual para o desenvolvimento de aplicações paralelas (contendo o *kit* de desenvolvimento C e *OpenMPI* pré-instalados) e criação de máquina virtual para o desenvolvimento de aplicações paralelas no *Google Cloud*. Disponível em <http://tiny.cc/material_apoio>;
  - Cadernos de desafios contendo teoria e desafios de programação, além das implementações correspondentes, para o conteúdo de *OpenMP, MPI e CUDA*. Disponível em <http://tiny.cc/cadernos_de_desafios>;

- Utilização da abordagem *Team Based Learning* aplicada ao ensino-aprendizado de programação paralela;
- Estudo sobre a utilização e impacto do uso de desafios de programação para o ensino, do ponto de vista do professor, e aprendizado, na visão do aluno, de programação paralela. Verificou-se, neste caso que, sob a perspectiva do professor, os desafios facilitam a correção e *feedback* aos alunos, quando aliado a um sistema de maratona de programação, embora gere custo adicional considerável para a preparação de aulas, com ou sem o uso de maratonas, e para manutenção da infraestrutura computacional, quando há o uso de maratonas de programação;
- Verificação que os desafios de programação paralela podem ser aplicados em aulas com diferentes metodologias de ensino (ativas ou tradicionais), com ou sem ferramentas de suporte a maratonas;
- Por fim, a maior contribuição vem ao encontro do aluno, ao estimular o desenvolvimento orientado aos casos de testes associados aos desafios, provendo um ambiente de competição saudável e permitindo que o aluno produza soluções melhores, mesmo já tendo submetido uma solução correta. As premiações, quando existentes e ainda que simbólicas, estimulam a participação dos alunos, contribuindo para o ambiente de competição saudável, já mencionado. O aluno sente sua evolução, o que permite a constante melhoria das soluções desenvolvidas. O aluno se sente motivado a continuar, superando-se em novos desafios do aprendizado.

6.4 Produção Científica

No intuito de tornar públicos os resultados obtidos por meio da realização deste trabalho, foram submetidos 3 *full papers* à conferência *Frontiers in Education (FIE)*, os quais foram aceitos após revisões.

Os 3 referidos estudos são: o uso de desafios de programação para a aprendizagem de programação paralela, que consiste nas contribuições da presente dissertação; o ensino de programação paralela para alunos iniciantes em computação, no qual houve participação do autor deste trabalho na execução dos experimentos; e o mapeamento sistemático sobre metodologias e recursos educacionais aplicados ao ensino e aprendizado de programação paralela, desenvolvido pelo autor deste trabalho, em conjunto de outros pesquisadores.

6.5 Limitações

As limitações deste trabalho podem ser relacionadas à subjetividade das correções visto que, independentemente do uso de sistemas de maratona de programação, ainda houve a validação das pontuações por diferentes professores humanos o que, naturalmente pode incorrer em inconsistências derivadas de falha ou visões distintas.

Por este motivo, foi aplicada uma métrica padronizada para o cálculo de qualidade de código que visa mitigar os efeitos de quaisquer parcialidades advinda dos corretores.

Além disso, as metodologias e recursos educacionais utilizados que, embora tenham sido variados nos experimentos e selecionados em função dos padrões e tendências identificados na literatura, podem influenciar os resultados obtidos.

Finalmente, a amostra selecionada dentro da população de todas as abordagens, métodos e técnicas de ensino-aprendizado, ainda é pequena e precisa ser expandida a novos alunos e docentes.

6.6 Trabalhos Futuros

- Analisar diferentes metodologias, além das consideradas neste trabalho;
- Estudar o impacto, usabilidade e funcionalidades de diferentes sistemas de maratona de programação em contextos diversos;
- Implementar ou adaptar um sistema de maratona de programação para fins de ensino;
- Adaptar os desafios de programação com base em problemas da indústria, visando aumentar a aplicabilidade e comprovação do modelo proposto em cenários realistas;
- Estudar o impacto a longo prazo do uso de desafios de programação.

REFERÊNCIAS

ABDULSALAM, A. A. Using model checking tool for teaching concurrent programming concepts. In: IEEE. **Innovations in Information Technology, 2009. IIT'09. International Conference on**. [S.l.], 2009. p. 215–219. Citado na página 76.

ACACIO, M. E.; CUENCA, J.; FERNÁNDEZ, L.; FERNÁNDEZ-PASCUAL, R.; CERVERA, J.; GIMÉNEZ, D.; GARRIDO, M. C.; LAGUNA, J. A. S.; GUILLÉN, J.; BENITO, J. A. P.; REQUENA, M.-E. An experience of early initiation to parallelism in the computing engineering degree at the University of Murcia, Spain. In: **Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012**. [S.l.: s.n.], 2012. p. 1289–1294. Citado na página 75.

ACM. **Programming Contest Control System**. 2015. Acessado em 07/03/2018. Disponível em: <http://pc2.ecs.csus.edu/pc2desc.html>. Citado nas páginas 47 e 48.

ACM/IEEE-CS Joint Task Force on Computing Curricula. **Computer Science Curricula 2013**. New York, NY, USA: ACM, 2013. Disponível em: <http://dx.doi.org/10.1145/2534860>. Citado nas páginas 53, 54, 56, 57, 60, 63 e 83.

ADAMS, J.; BROWN, R.; SHOOP, E. Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to cs undergraduates. In: IEEE. **Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International**. [S.l.], 2013. p. 1244–1251. Citado na página 75.

ADAMS, J. C. Injecting parallel computing into cs2. In: ACM. **Proceedings of the 45th ACM technical symposium on Computer science education**. [S.l.], 2014. p. 277–282. Citado na página 75.

____. Patternlets: A teaching tool for introducing students to parallel design patterns. In: IEEE. **2015 IEEE International Parallel and Distributed Processing Symposium Workshop**. [S.l.], 2015. p. 752–759. Citado na página 76.

ADAMS, J. C.; CRAIN, P. A.; DILLEY, C. P.; HAZLETT, C. D.; KONING, E. R.; NELESEN, S. M.; UNGER, J. B.; STEL, M. B. V. Tsgl: A tool for visualizing multithreaded behavior. **Journal of Parallel and Distributed Computing**, Elsevier, v. 118, p. 233–246, 2018. Citado na página 76.

ALMASI, G. S.; GOTTLIEB, A. **Highly Parallel Computing**. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1989. Citado na página 27.

ALMEIDA, F.; CUENCA, J.; FERN'NDEZ-PASCUAL, R.; GIMÉNEZ, D.; BENITO, J. A. P. The spanish parallel programming contests and its use as an educational resource. In: IEEE. **Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International**. [S.l.], 2012. p. 1303–1306. Citado nas páginas 28, 45, 46, 62, 76, 77, 79 e 136.

ANTHONY, E. M. Approach, method, and technique. **ELT Journal**, XVII, n. 2, p. 63–67, 1963. Citado nas páginas 34, 35 e 38.

ARMITAGE, P.; BERRY, G.; MATTHEWS, J. N. S. **Statistical methods in medical research**. [S.l.]: John Wiley & Sons, 2008. Citado na página 83.

ARROYO, M. Teaching parallel and distributed computing to undergraduate computer science students. In: IEEE. **Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International**. [S.l.], 2013. p. 1297–1303. Citado na página 75.

AYGUADÉ, E.; BADIA, R. M.; JIMÉNEZ, D.; HERRERO, J. R.; LABARTA, J.; SUBOTIC, V.; UTRERA, G. Tareador: a tool to unveil parallelization strategies at undergraduate level. In: ACM. **Proceedings of the Workshop on Computer Architecture Education**. [S.l.], 2015. p. 1. Citado nas páginas 75 e 76.

AZIZ, M.; CHI, H.; TIBREWAL, A.; GROSSMAN, M.; SARKAR, V. Auto-grading for parallel programs. In: **Proceedings of the Workshop on Education for High-Performance Computing**. [S.l.: s.n.], 2015. p. 1–8. Citado na página 76.

BACHIEGA, N. G.; SOUZA, P. S. L.; BRUSCHI, S.; DO, S.; SOUZA, R. S. D. Mapeamento sistemático do ensino teórico e prático de programação paralela. In: **VI Congresso Brasileiro de Informática na Educação**. [s.n.], 2017. p. 1089–1098. Disponível em: <http://www.br-ie.org/pub/index.php/wcbie/article/viewFile/7498/5293>. Citado nas páginas 62 e 65.

BERNABÉ, G.; CUENCA, J.; GARCÍA, L.-P.; GIMÉNEZ, D.; RIVAS-GOMEZ, S. A high performance computing course guided by the lu factorization. **Procedia Computer Science**, Elsevier, v. 29, p. 1446–1457, 2014. Citado na página 75.

BI, Y.; BEIDLER, J. A Visual Tool for Teaching Multithreading in Java. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 22, n. 6, p. 156–163, 2007. ISSN 1937-4771. Citado na página 76.

BISHOP, J. L.; VERLEGER, M. A. The flipped classroom: A survey of the research. In: **ASEE National Conference Proceedings, Atlanta, GA**. [S.l.: s.n.], 2013. v. 30, n. 9, p. 1–18. Citado na página 44.

BOCA. **BOCA Online Contest Administrator**. 2010. Acessado em 07/03/2018. Disponível em: <https://www.ime.usp.br/~cassio/boca>. Citado na página 48.

BOERSEN, R.; PHILLIPPS, M. Programming contests: Two innovative models from new zealand. **text and description**, p. 1–63, 2006. Citado na página 46.

BONWELL, C. C.; EISON, J. A. **Active learning: creating excitement in the classroom**. [S.l.]: George Washington University, ERIC Clearinghouse on Higher Education, 1991. Citado nas páginas 36, 37, 38, 39 e 44.

BOSSCHERE, K. O. D. Edulan, a tool for teaching synchronization. **Microprocessing and Microprogramming**, Elsevier, v. 27, n. 1-5, p. 819–825, 1989. Citado na página 76.

BOUD, D.; COHEN, R.; SAMPSON, J. **Peer learning in higher education: Learning from and with each other**. [S.l.]: Routledge, 2014. Citado na página 43.

BOURGOIN, M.; CHAILLOUX, E.; LAMOTTE, J.-L. Efficient abstractions for gpgpu programming. **International Journal of Parallel Programming**, v. 42, n. 4, p. 583–600, 2014. Citado nas páginas 28 e 29.

BOYLE, R. A. Employing active-learning techniques and metacognition in law school: Shifting energy from professor to student. **U. Det. Mercy L. Rev.**, HeinOnline, v. 81, p. 1, 2003. Citado na página 37.

BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. **Journal of systems and software**, Elsevier, v. 80, n. 4, p. 571–583, 2007. Citado nas páginas 65 e 67.

BRISCOE, G.; MULLIGAN, C. **Digital innovation: The hackathon phenomenon**. 2014. Citado na página 46.

BROWN, C. M.; LU, Y.-H.; MIDKIFF, S. Introducing parallel programming in undergraduate curriculum. In: **Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium Workshops and PhD Forum, IPDPSW 2013**. [S.l.: s.n.], 2013. p. 1269–1274. Citado na página 75.

BROWN, R.; SHOOP, E. CSinParallel and synergy for rapid incremental addition of PDC into CS curricula. In: **Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012**. [S.l.: s.n.], 2012. p. 1329–1334. Citado na página 75.

BROWN, R. A.; ADAMS, J. C.; FERNER, C.; SHOOP, E.; WILKINSON, A. B. Teaching parallel design patterns to undergraduates in computer science. In: ACM. **Proceedings of the 45th ACM technical symposium on Computer science education**. [S.l.], 2014. p. 547–548. Citado na página 75.

BRUFFEE, K. A. **Collaborative learning higher education, interdependence, and the authority of knowledge**. [S.l.]: Johns Hopkins University Press, 1999. Citado na página 42.

BUI, P.; BOETTCHER, T.; JAEGER, N.; WESTPHAL, J. Using clusters in undergraduate research: Distributed animation rendering, photo processing, and image transcoding. In: IEEE. **2013 IEEE International Conference on Cluster Computing (CLUSTER)**. [S.l.], 2013. p. 1–8. Citado na página 76.

BURKHART, H. Parallel programming using public domain software. In: ACM. **ACM SIGCSE Bulletin**. [S.l.], 1997. v. 29, n. 1, p. 224–228. Citado na página 76.

BURKHART, H.; GUERRERA, D.; MAFFIA, A. Trusted high-performance computing in the classroom. In: IEEE PRESS. **Proceedings of the Workshop on Education for High-Performance Computing**. [S.l.], 2014. p. 27–33. Citado na página 75.

BURTSCHER, M.; PENG, W.; QASEM, A.; SHI, H.; TAMIR, D.; THIRY, H. A module-based approach to adopting the 2013 acm curricular recommendations on parallel computing. In: ACM. **Proceedings of the 46th ACM Technical Symposium on Computer Science Education**. [S.l.], 2015. p. 36–41. Citado na página 75.

CACHO, C. E.; SOUZA, P. S.; BRUSCHI, S. M.; BARBOSA, E. F.; TIOSSO, F. An interactive approach for the teaching of virtual memory using open educational resources. In: ACM. **Proceedings of the 31st Annual ACM Symposium on Applied Computing**. [S.l.], 2016. p. 225–231. Citado nas páginas 28 e 29.

CAPEL, M. I.; TOMEU, A. J.; SALGUERO, A. G. A set of patterns for concurrent and parallel programming teaching. In: SPRINGER. **European Conference on Parallel Processing**. [S.l.], 2017. p. 203–215. Citado na página 75.

____. Teaching concurrent and parallel programming by patterns: An interactive ict approach. **Journal of Parallel and Distributed Computing**, Elsevier, v. 105, p. 42–52, 2017. Citado na página 75.

CAPRETTI, G.; LAGANA, M. R.; RICCI, L.; CASTELLUCCI, R.; PURI, S. Orespics: a friendly environment to learn cluster programming. In: IEEE. **Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on**. [S.l.], 2001. p. 498–505. Citado na página 76.

CARR, S.; CHEN, P.; JOZWOWSKI, T. R.; MAYO, J.; SHENE, C.-K. Channels, Visualization, and Topology Editor. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 34, n. 3, p. 106–110, 2002. ISSN 0097-8418. Citado na página 76.

CARR, S.; FANG, C.; JOZWOWSKI, T.; MAYO, J.; SHENE, C.-K. Concurrent mentor: A visualization system for distributed programming education. In: **PDPTA**. [S.l.: s.n.], 2003. p. 1676–1682. Citado na página 76.

CARR, S.; MAYO, J.; SHENE, C.-K. Threadmentor: a pedagogical tool for multithreaded programming. **Journal on Educational Resources in Computing (JERIC)**, ACM, v. 3, n. 1, p. 1, 2003. Citado na página 76.

CARRO, M.; HERRANZ, Á.; MARINO, J. A model-driven approach to teaching concurrency. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 13, n. 1, p. 5, 2013. Citado na página 75.

CAZDEN, C. B. **Classroom discourse: The language of teaching and learning.** [S.l.]: ERIC, 1988. Citado nas páginas 33 e 34.

CHAUDHURY, B.; VARMA, A.; KESWANI, Y.; BHATNAGAR, Y.; PARIKH, S. Let's hpc: A web-based platform to aid parallel, distributed and high performance computing education. **Journal of Parallel and Distributed Computing**, Elsevier, v. 118, p. 213–232, 2018. Citado na página 76.

CHEN, J.; SHEN, L.; YIN, J.; ZHANG, C. Parallel programming course development based on parallel computational thinking. In: **Proceedings of ACM Turing Celebration Conference-China**. [S.l.: s.n.], 2018. p. 103–109. Citado na página 75.

CHICKERING, A. W.; GAMSON, Z. F. Development and adaptations of the seven principles for good practice in undergraduate education. **New directions for teaching and learning**, Wiley Online Library, v. 1999, n. 80, p. 75–81, 1999. Citado na página 36.

COMBÉFIS, S.; WAUTELET, J. Programming trainings and informatics teaching through online contests. **Olympiads in Informatics**, v. 8, 2014. Citado na página 46.

Concurso de Programación Paralela. **VII Concurso de Programación Paralela: 2017**. 2017. Acessado em 07/03/2018. Disponível em: <http://luna.inf.um.es/2017/>. Citado na página 50.

CUENCA, J.; GIMéNEZ, D. A parallel programming course based on an execution time-energy consumption optimization problem. In: **2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2016. p. 996–1003. Citado nas páginas 37 e 75.

DANNER, A.; NEWHALL, T.; WEBB, K. C. Paravis: A library for visualizing and debugging parallel applications. In: IEEE. **2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.], 2019. p. 326–333. Citado na página 76.

DAVIES, G. Teaching concurrent programming with pascal-fc. **ACM SIGCSE Bulletin**, ACM, v. 22, n. 2, p. 38–41, 1990. Citado na página 76.

De Freitas, H. C. Method for teaching parallelism on heterogeneous many-core processors using research projects. In: **Proceedings - Frontiers in Education Conference, FIE**. [S.l.: s.n.], 2013. p. 108–113. Citado nas páginas 75 e 77.

DEWEY, J. The class room teacher. **Science Education**, Wiley Online Library, v. 8, n. 3, p. 463–472, 1924. Citado nas páginas 39 e 40.

DICHEV, C.; DICHEVA, D. Open educational resources in computer science teaching. In: ACM. **Proceedings of the 43rd ACM technical symposium on Computer Science Education**. [S.l.], 2012. p. 619–624. Citado na página 28.

DILLENBOURG, P. What do you mean by collaborative leraning? In: DILLENBOURG, P. (Ed.). **Collaborative-learning: Cognitive and Computational Approaches**. Oxford: Elsevier, 1999. cap. 1, p. 1–19. Citado na página 42.

DOLGOPOLOVAS, V.; DAGIENĖ, V.; MINKEVIČIUS, S.; SAKALAUSKAS, L. Teaching scientific computing: A model-centered approach to pipeline and parallel programming with c. **Scientific Programming**, Hindawi Publishing Corp., v. 2015, p. 11, 2015. Citado na página 75.

DUCH, B. J.; GROH, S. E.; ALLEN, D. E. **The power of problem-based learning: a practical"how to"for teaching undergraduate courses in any discipline**. [S.l.]: Stylus Publishing, LLC., 2001. Citado na página 39.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and software technology**, Elsevier, v. 50, n. 9, p. 833–859, 2008. Citado nas páginas 69 e 74.

D'ANTONI, S. Open educational resources: the way forward. **Open Educational Resources Conversations in cyberspace**, v. 11, p. 161, 2009. Citado na página 28.

EIJKHOUT, V. Teaching mpi from mental models. In: IEEE. **Education for High-Performance Computing (EduHPC), 2016 Workshop on**. [S.l.], 2016. p. 14–18. Citado na página 75.

ERNST, D. J.; STEVENSON, D. E. Concurrent cs: Preparing students for a multicore world. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 40, n. 3, p. 230–234, jun. 2008. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/1597849.1384333>. Citado na página 53.

FAUST, J. L.; PAULSON, D. R. Active learning in the college classroom. **Journal on excellence in college teaching**, v. 3, n. 2, p. 3–24, 1998. Citado na página 36.

FELDHAUSEN, R.; BELL, S.; 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**. New York, NY, USA: ACM, 2014. (XSEDE '14), p. 75:1—-75:7. ISBN 978-1-4503-2893-7. Citado na página 75.

FENG, A.; TILEVICH, E.; FENG, W.-C. Block-based programming abstractions for explicit parallel computing. In: **Proceedings - 2015 IEEE Blocks and Beyond Workshop, Blocks and Beyond 2015**. [S.l.: s.n.], 2015. p. 71–75. Citado na página 75.

FERNER, C.; WILKINSON, B.; HEATH, B. Toward using higher-level abstractions to teach parallel computing. In: IEEE. **Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International**. [S.l.], 2013. p. 1291–1296. Citado na página 75.

____. Using patterns to teach parallel computing. In: IEEE. **2014 IEEE International Parallel & Distributed Processing Symposium Workshops**. [S.l.], 2014. p. 1106–1113. Citado na página 75.

FERREIRA, D. C. e C. E. Boca: um sistema de apoio a competiçoes de programaçao (boca: A support system for programming contests). In: **Anais do Congresso da SBC-Workshop de Educaçao em Computaç ao**. [S.l.: s.n.], 2004. p. 885–895. Citado nas páginas 47 e 48.

FIORE, S.; BAKHOUYA, M.; SMARI, W. W. On the road to exascale : Advances in High Performance Computing and Simulations — An overview and editorial. **Future Generation Computer Systems**, Elsevier B.V., v. 82, p. 450–458, 2018. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2018.01.034>. Citado na página 27.

FOLEY, S. S.; HURSEY, J. OnRamp to Parallel and Distributed Computing. In: **Proceedings of the Workshop on Education for High-Performance Computing**. New York, NY, USA: ACM, 2015. (EduHPC '15), p. 1:1—-1:8. ISBN 978-1-4503-3961-2. Citado na página 76.

FOLEY, S. S.; KOEPKE, D.; RAGATZ, J.; BREHM, C.; REGINA, J.; HURSEY, J. Onramp: A web-portal for teaching parallel and distributed computing. **Journal of Parallel and Distributed Computing**, Elsevier, v. 105, p. 138–149, 2017. Citado na página 76.

FREEMAN, S.; EDDY, S. L.; MCDONOUGH, M.; SMITH, M. K.; OKOROAFOR, N.; JORDT, H.; WENDEROTH, M. P. Active learning increases student performance in science, engineering, and mathematics. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 111, n. 23, p. 8410–8415, 2014. Citado na página 37.

FRESNO, J.; ORTEGA-ARRANZ, A.; ORTEGA-ARRANZ, H.; GONZALEZ-ESCRIBANO, A.; LLANOS, D. R. **Applying gamification in a parallel programming course**. [S.l.: s.n.], 2016. 108–130 p. Citado nas páginas 75 e 77.

GIACAMAN, N.; SINNEN, O. EA: Research-infused teaching of parallel programming concepts for undergraduate software engineering students. In: **Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS**. [S.l.: s.n.], 2014. p. 1099–1105. Citado na página 75.

GIMÉNEZ, D. A practical parallel programming course based on problems of the Spanish parallel programming contest. In: **Procedia Computer Science**. [S.l.: s.n.], 2016. v. 80, p. 1978–1988. Citado nas páginas 28, 46, 50, 62, 75, 76, 78, 79 e 136.

GOODLAD, S.; HIRST, B. **Peer Tutoring: A Guide to Learning by Teaching**. [S.l.]: Kogan Page, 1989. ISBN 9781850917779. Citado na página 43.

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: ACM, 2011. (SIGCSE '11), p. 435–440. ISBN 978-1-4503-0500-6. Citado na página 75.

GROSSMAN, M.; AZIZ, M.; CHI, H.; TIBREWAL, A.; IMAM, S.; SARKAR, V. Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level. **Journal of Parallel and Distributed Computing**, Elsevier, v. 105, p. 18–30, 2017. Citado nas páginas 75 e 76.

HAGER, G.; WELLEIN, G. **Introduction to high performance computing for scientists and engineers**. [S.l.]: CRC Press, 2010. Citado na página 27.

HILLS, H. **Team-based learning**. [S.l.]: Gower Publishing, Ltd., 2001. Citado na página 43.

HUOTARI, K.; HAMARI, J. Defining gamification: a service marketing perspective. In: ACM. **Proceeding of the 16th international academic MindTrek conference**. [S.l.], 2012. p. 17–22. Citado na página 44.

IMAM, S.; SARKAR, V. Habanero-Java Library: A Java 8 Framework for Multicore Programming. In: **Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools**. New York, NY, USA: ACM, 2014. (PPPJ '14), p. 75–86. ISBN 978-1-4503-2926-2. Citado na página 76.

INUWA, I. Perceptions and attitudes of first-year medical students on a modified team-based learning (tbl) strategy in anatomy. v. 12, p. 336–43, 08 2012. Citado na página 44.

IPARRAGUIRRE, J.; FRIEDRICH, G. R.; COPPO, R. J. Lessons Learned after the Introduction of Parallel and Distributed Computing Concepts into ECE Undergraduate Curricula at UTN-Bah #x00ED;a Blanca Argentina. In: **2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum**. [S.l.: s.n.], 2012. p. 1317–1320. Citado na página 75.

JOINER, D. A.; GRAY, P.; MURPHY, T.; PECK, C. Teaching parallel computing to science faculty: best practices and common pitfalls. In: ACM. **Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming**. [S.l.], 2006. p. 239–246. Citado na página 75.

KAILA, E. Utilizing educational technology in computer science and programming courses: theory and practice. Turku Centre for Computer Science, 2018. Citado na página 37.

KELLER, J. **Motivational Design for Learning and Performance: The ARCS Model Approach**. [S.l.: s.n.], 2010. 1-353 p. Citado nas páginas 83, 102, 108, 115, 128 e 137.

KHERA, V.; ASTRACHAN, O.; KOTZ, D. The internet programming contest: a report and philosophy. **ACM SIGCSE Bulletin**, ACM, v. 25, n. 1, p. 48–52, 1993. Citado nas páginas 45, 46 e 47.

KITCHEN, A. T.; SCHALLER, N. C.; TYMANN, P. T. Game Playing As a Technique for Teaching Parallel Computing Concepts. **SIGCSE Bull.**, ACM, New York, NY, USA, v. 24, n. 3, p. 35–38, 1992. ISSN 0097-8418. Citado na página 75.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004. Citado na página 65.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Citado nas páginas 65, 68 e 73.

KO, Y.; BURGSTALLER, B.; 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**. New York, NY, USA: ACM, 2013. (SIGCSE '13), p. 415–420. ISBN 978-1-4503-1868-6. Citado na página 75.

LAGE, M. J.; PLATT, G. J.; TREGLIA, M. Inverting the classroom: A gateway to creating an inclusive learning environment. **The Journal of Economic Education**, Taylor & Francis, v. 31, n. 1, p. 30–43, 2000. Citado na página 43.

LARMER, J.; MERGENDOLLER, J.; BOSS, S. **Setting the standard for project based learning**. [S.l.]: ASCD, 2015. Citado na página 41.

LARMER, J.; MERGENDOLLER, J. R. Essentials for project-based learning. **Educational leadership**, v. 68, n. 1, p. 34–37, 2012. Citado na página 41.

LEAL, J. P.; SILVA, F. Mooshak: A web-based multi-site programming contest system. **Software: Practice and Experience**, Wiley Online Library, v. 33, n. 6, p. 567–581, 2003. Citado nas páginas 47, 49 e 77.

LEVANDOWSKI, B.; PEROULI, D.; BRYLOW, D. Using embedded xinu and the raspberry pi 3 to teach parallel computing in assembly programming. In: IEEE. **2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.], 2019. p. 334–341. Citado na página 75.

LI, J.; GUO, W.; ZHENG, H. An undergraduate parallel and distributed computing course in multi-core era. In: **Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008**. [S.l.: s.n.], 2008. p. 2412–2416. Citado na página 75.

LIN, S.; TATAR, D. Encouraging Parallel Thinking Through Explicit Coordination Modeling. In: **Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2011. (SIGCSE '11), p. 441–446. ISBN 978-1-4503-0500-6. Citado na página 75.

LIU, J. 20 Years of Teaching Parallel Processing to Computer Science Seniors. In: **Proceedings of EduHPC 2016: Workshop on Education for High-Performance Computing - Held in conjunction with SC 2016: The International Conference for High Performance Computing, Networking, Storage and Analysis**. [S.l.: s.n.], 2017. p. 7–13. Citado na página 75.

LIU, P.; LEE, G. C.; LEE, J.-K.; LIN, C.-Y. Innovative System and Application Curriculum on Multicore Systems. In: **Proceedings of the 6th Workshop on Embedded Systems Education**. New York, NY, USA: ACM, 2011. (WESE '11), p. 25–31. ISBN 978-1-4503-1046-8. Citado na página 75.

LOPES, R. **A Formação Interdisciplinar dos Professores de Ciências da Natureza Para a Integração Curricular Através da Aprendizagem Baseada em Problemas**. 2017. Citado na página 40.

LUPO, C.; WOOD, Z. J.; 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**. New York, NY, USA: ACM, 2012. (SIGCSE '12), p. 523–528. ISBN 978-1-4503-1098-7. Citado nas páginas 75 e 77.

MANEV, K.; KELEVEDJIEV, E.; KAPRALOV, S. Programming contests for school students in bulgaria. **Olympiads in Informatics**, v. 1, p. 112–123, 2007. Citado na página 46.

MANOGARAN, E. ACt-PBL: An adaptive approach to teach multi-core computing in university education. In: **Proceedings - 2013 IEEE 5th International Conference on Technology for Education, T4E 2013**. [S.l.: s.n.], 2013. p. 19–23. Citado na página 75.

MARKHAM, T. Project based learning a bridge just far enough. **Teacher Librarian**, EL Kurdyla Publishing LLC, v. 39, n. 2, p. 38, 2011. Citado na página 40.

MCCARTHY, J. P.; ANDERSON, L. Active learning techniques versus traditional teaching styles: Two experiments from history and political science. **Innovative Higher Education**, v. 24, n. 4, p. 279–294, Jun 2000. ISSN 1573-1758. Citado na página 37.

MCCONNELL, J. J. Active learning and its use in computer science. **ACM SIGCSE Bulletin**, ACM, v. 28, n. SI, p. 52–54, 1996. Citado na página 37.

MELLOR-CRUMMEY, J.; GROPP, W.; HERLIHY, M. Teaching parallel programming: a roundtable discussion. **XRDS: Crossroads, The ACM Magazine for Students**, ACM, v. 17, n. 1, p. 28–30, 2010. Citado na página 62.

MICHAELSEN, L. K.; SWEET, M. Team-based learning. **New directions for teaching and learning**, Wiley Online Library, v. 2011, n. 128, p. 41–51, 2011. Citado na página 43.

MIZUKAMI, M. d. G. N. **Ensino: as abordagens do processo**. [S.l.: s.n.], 1986. Citado nas páginas 34, 35 e 37.

MOORE, S. V.; DUNLOP, S. R. A flipped classroom approach to teaching concurrency and parallelism. In: **2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2016. p. 987–995. Citado nas páginas 37 e 75.

MOOSHAK. **Mooshak Home Page**. 2016. Acessado em 07/03/2018. Disponível em: <https://mooshak.dcc.fc.up.pt/>. Citado na página 49.

MURESANO, R.; REXACHS, D.; LUQUE, E. International Conference on Computational Science , ICCS 2010 Learning parallel programming : a challenge for university students. **Procedia Computer Science**, Elsevier, v. 1, n. 1, p. 875–883, 2012. ISSN 1877-0509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2010.04.096>. Citado nas páginas 28, 61 e 75.

PARK, E. L.; CHOI, B. K. Transformation of classroom spaces: Traditional versus active learning classroom in colleges. **Higher Education**, Springer, v. 68, n. 5, p. 749–771, 2014. Citado nas páginas 36 e 37.

PARMELEE, D.; MICHAELSEN, L. K.; COOK, S.; HUDES, P. D. Team-based learning: a practical guide: Amee guide no. 65. **Medical teacher**, Taylor & Francis, v. 34, n. 5, p. e275–e287, 2012. Citado nas páginas 86, 88 e 89.

PHAM, M. T.; NGUYEN, T. B. The domjudge based online judge system with plagiarism detection. In: IEEE. **2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)**. [S.l.], 2019. p. 1–6. Citado na página 50.

POPOVIĆ, M.; VLADIMIR, K.; ŠILIĆ, M. Application of social game context to teaching mutual exclusion. **Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije**, KoREMA-Hrvatsko društvo za komunikacije, računarstvo, elektroniku, mjerenja . . . , v. 59, n. 2, p. 208–219, 2018. Citado na página 76.

PRASAD, S. K.; CHTCHELKANOVA, A.; GUPTA, A.; ROSENBERG, A.; SUSSMAN, A. Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: Core topics for undergraduates. In: **Proceedings of the 45th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2014. (SIGCSE '14). ISBN 978-1-4503-2605-6. Disponível em: <https://grid.cs.gsu.edu/~tcpp/curriculum/>. Acesso em: 01/02/2018. Citado nas páginas 60, 61 e 83.

PRASAD, S. K.; CHTCHELKANOVA, A. Y.; DAS, S. K.; DEHNE, F.; GOUDA, M. G.; GUPTA, A.; JAJA, J.; KANT, K.; SALLE, A. L.; LEBLANC, R. *et al.* Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: core topics for undergraduates. In: **SIGCSE**. [S.l.: s.n.], 2011. v. 11, p. 617–618. Citado na página 61.

PRAUN, C. von. Parallel programming: design of an overview class. In: **Proceedings of the 2011 ACM SIGPLAN X10 Workshop**. [S.l.: s.n.], 2011. p. 1–6. Citado na página 75.

PREGUNTA, L. A estratégia pico para a construção da pergunta de pesquisa e busca de evidências. **Rev Latino-am Enfermagem**, SciELO Brasil, v. 15, n. 3, 2007. Citado na página 66.

RADENSKI, A. Integrating data-intensive cloud computing with multicores and clusters in an hpc course. In: **Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education**. [S.l.: s.n.], 2012. p. 69–74. Citado nas páginas 75 e 76.

RAGUE, B. Measuring CS1 perceptions of parallelism. In: **Proceedings - Frontiers in Education Conference, FIE**. [S.l.: s.n.], 2011. Citado na página 75.

RAGUE, B. W. Exploring concurrency using the parallel analysis tool. In: **SIGCSE'12 - Proceedings of the 43rd ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.], 2012. p. 511–516. Citado na página 76.

RIVOIRE, S. A breadth-first course in multicore and manycore programming. In: **SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education**. [S.l.: s.n.], 2010. p. 214–218. Citado na página 75.

SADOWSKI, C.; BALL, T.; BISHOP, J.; BURCKHARDT, S.; GOPALAKRISHNAN, G.; MAYO, J.; MUSUVATHI, M.; QADEER, S.; TOUB, S. Practical parallel and concurrent programming. In: **Proceedings of the 42nd ACM technical symposium on Computer science education**. [S.l.: s.n.], 2011. p. 189–194. Citado nas páginas 75 e 76.

SAPARKHOJAYEV, N. Effective teaching of "parallel computing"course by using microlearning technique. **World Applied Sciences Journal**, v. 28, n. 6, p. 842–846, 2013. Citado na página 75.

SARKAR, V.; GROSSMAN, M.; BUDIMLIĆ, Z.; IMAM, S. Preparing an online java parallel computing course. In: IEEE. **2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.], 2017. p. 360–366. Citado na página 76.

SHAMSI, J. A.; DURRANI, N. M.; KAFI, N. Novelties in teaching high performance computing. In: IEEE. **Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International**. [S.l.], 2015. p. 772–778. Citado nas páginas 28, 29, 61 e 75.

SHILOV, N. V.; YI, K. Engaging students with theory through acm collegiate programming contest. **Communications of the ACM**, ACM, v. 45, n. 9, p. 98–101, 2002. Citado na página 46.

SHOVEIN, J.; HUSTON, C.; FOX, S.; DAMAZO, B. Challenging traditional teaching and learning paradigms: Online learning and emancipatory teaching. **Nursing Education Perspectives**, LWW, v. 26, n. 6, p. 340–343, 2005. Citado na página 35.

SILBERMAN, M. **Active Learning: 101 Strategies to Teach Any Subject**. [S.l.]: Allyn and Bacon, 1996. ISBN 9780205178667. Citado na página 38.

SIMON, B.; ANDERSON, R.; HOYER, C.; SU, J. Preliminary experiences with a tablet pc based system to support active learning in computer science courses. **ACM SIGCSE Bulletin**, ACM, v. 36, n. 3, p. 213–217, 2004. Citado na página 37.

SKIENA, S. S.; REVILLA, M. A. **Programming challenges: The programming contest training manual**. [S.l.]: Springer Science & Business Media, 2006. Citado na página 45.

SKOPIN, I. N. **Early learning in parallel programming**. [S.l.: s.n.], 2014. 219–229 p. Citado na página 75.

STRAZDINS, P. E. Experiences in teaching a specialty multicore computing course. In: **Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012**. [S.l.: s.n.], 2012. p. 1283–1288. Citado na página 75.

STUFFLEBEAM, D. L.; CORYN, C. L. **Evaluation theory, models, and applications**. [S.l.]: John Wiley & Sons, 2014. v. 50. Citado na página 137.

TOMEU-HARDASMAL, A. J.; SALGUERO, A. G.; CAPEL, M. I. Integration of ICT in concurrent and parallel programming lectures. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 9523, p. 114–124, 2015. Citado na página 75.

TORBERT, S.; VISHKIN, U.; TZUR, R.; ELLISON, D. J. 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**. New York, NY, USA: ACM, 2010. (SIGCSE '10), p. 290–294. ISBN 978-1-4503-0006-3. Citado na página 75.

TSIOPOULOS, L.; JOHKIO, F. A.; GEORGAKARAKOS, G.; DAHLIN, A.; LILIUS, J. 1Teaching many-core programming. In: **10th European Workshop on Microelectronics Education (EWME)**. [S.l.: s.n.], 2014. p. 7–10. Citado na página 75.

UL-ABDIN, Z.; SVENSSON, B. Towards teaching embedded parallel computing: An analytical approach. In: ACM. **Proceedings of the Workshop on Computer Architecture Education**. [S.l.], 2015. p. 8. Citado na página 75.

VALENTINE, D. HPC/PDC Immunization in the Introductory Computer Science Sequence. In: **Proceedings of the Workshop on Education for High-Performance Computing**. Piscataway, NJ, USA: IEEE Press, 2014. (EduHPC '14), p. 9–14. ISBN 978-1-4799-7021-6. Citado na página 75.

VALLS-VARGAS, J.; ZHU, J.; ONTAÑÓN, S. Graph grammar-based controllable generation of puzzles for a learning game about parallel programming. In: **Proceedings of the 12th International Conference on the Foundations of Digital Games**. [S.l.: s.n.], 2017. p. 1–10. Citado na página 76.

WALPOLE, R. E.; MYERS, R. H.; MYERS, S. L.; YE, K. **Probability and statistics for engineers and scientists**. [S.l.]: Macmillan New York, 1993. v. 5. Citado na página 133.

WHITE, B.; ROBINSON, M.; MITCHELL, C.; MACHE, J. Using the n-body problem to engage undergraduates in parallel programming. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... **Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)**. [S.l.], 2012. p. 1. Citado na página 75.

WOHLIN, C.; RUNESON, P.; HöST, M.; OHLSSON, M.; REGNELL, B.; WESSLéN, A. Experimentation in software engineering. In: ____. [S.l.: s.n.], 2012. p. 123–151. ISBN 978-3-642-29043-5. Citado nas páginas 31 e 83.

YAZICI, A.; MISHRA, A.; KARAKAYA, Z. Teaching parallel computing concepts using real-life applications. **International Journal of Engineering Education**, TEMPUS PUBLICATIONS IJEE, ROSSMORE,, DURRUS, BANTRY, COUNTY CORK 00000, IRELAND, v. 32, n. 2, p. 772–781, 2016. Citado na página 75.

ZAKHAROVA, I.; ZAKHAROV, A. Key issues of low-level parallel programming in theindividual projects for graduate students. **The International journal of engineering education**, Instituto de Relaciones Internacionales"Daza de Valdes", v. 34, n. 4, p. 1250–1260, 2018. Citado na página 75.

ZARESTKY, J.; BANGERTH, W. Teaching high performance computing: Lessons from a flipped classroom, project-based course on finite element methods. In: IEEE PRESS. **Proceedings of the Workshop on Education for High-Performance Computing**. [S.l.], 2014. p. 34–41. Citado na página 75.


ICMC
USP
SÃO CARLOS