Computational Science and Engineering Education in Different Countries
IEEE Computer Society Has You Covered!

WORLD-CLASS CONFERENCES — Stay ahead of the curve by attending one of our 200+ globally recognized conferences.

DIGITAL LIBRARY — Easily access over 780k articles covering world-class peer-reviewed content in the IEEE Computer Society Digital Library.

CALLS FOR PAPERS — Discover opportunities to write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog and its range of offerings.

ADVANCE YOUR CAREER — Search the new positions posted in the IEEE Computer Society Jobs Board.

NETWORK — Make connections that count by participating in local Region, Section, and Chapter activities.

Explore all of the member benefits at www.computer.org today!
Your Homework Assignment:
Nargess Memarsadeghi, NASA Goddard Space Flight Center, nargess.memarsadeghi@nasa.gov

IEEE ANTENNAS & PROPAGATION SOCIETY LIAISON
Don Wilton, University of Houston, wilton@uh.edu

IEEE SIGNAL PROCESSING SOCIETY LIAISON
Mritunjyot Chakraborty, Indian Institute of Technology, mritunjyot@iitkgp.ernet.in

CS MAGAZINE OPERATIONS COMMITTEE
Diomidis Spinellis (Chair), Lorena Barba, Irena Bojanova, Shu-Chung Chen, Gerardo Con Diaz, Lizy K. John, Marc Langheinrich, Torsten Moller, David Nicl, Ipek Ozkaya, George Pallis, VS Subrahmanian, Jeffrey Voas

CS PUBLICATIONS BOARD

COMPUTING IN SCIENCE & ENGINEERING STAFF
Journals Production Manager: Shannon Campos, shannon.campos@ieee.org
Cover Design: Janet Dudar
Peer Review Administrator: cise@computer.org
Publications Portfolio Manager: Carrie Clark
Publisher: Robin Baldwin
Senior Advertising Coordinator: Debbie Sims
IEEE Computer Society Executive Director: Melissa Russell

IEEE PUBLISHING OPERATIONS
Senior Director, Publishing Operations: Dawn Melley
Director, Editorial Services: Kevin Lisankie
Director, Production Services: Peter M. Tuohey
Associate Director, Editorial Services: Jeffrey E. Cichocki
Associate Director, Information Conversion and Editorial Support: Neelam Khinivasara
Senior Art Director: Janet Dudar
Manager, Journals Production: Katie Sullivan

COMPUTER SOCIETY OFFICE
COMPUTING IN SCIENCE & ENGINEERING
c/o IEEE Computer Society
10662 Los Vaqueros Circle, Los Alamitos, CA 90720 USA
Phone +1 714 821 8380; Fax +1 714 821 4010
Website: www.computer.org/cise

EDITOR-IN-CHIEF
Lorena A. Barba, The George Washington University, labarba@gwu.edu

ASSOCIATE EDITORS-IN-CHIEF
Jeffrey Carver, University of Alabama, carver@cs.ua.edu
Steven Gottlieb, Indiana University, sg@indiana.edu
Douglas E. Post, Carnegie Mellon University, post@ieee.org
Barry I. Schneider, NIST, bis@nist.gov
John West, University of Texas, Austin, john@tacc.utexas.edu

DEPARTMENT EDITORS
Computing Prescriptions: Francis Sullivan, IDA/Center for Computing Sciences, fran@super.org
Computer Simulations: Barry I. Schneider, NIST, bis@nist.gov, and Gabriel A. Wainer, Carleton University, gwainer@sce.carleton.ca
Data Track: Manish Parashar, Rutgers University, parashar@rutgers.edu, and George K. Thiruvathukal, Loyola University Chicago, gkt@cs.luc.edu
Diversity and Inclusion: Mary Ann Leung, Sustainable Horizons Institute, mleung@sustainablehorizons.org
Education: Sharon Broude Geva, University of Michigan, sgeva@umich.edu, and Dirck Colby, Michigan State University, colbyd@msu.edu
Leadership Computing: James J. Hack, ORNL, jhack@ornl.gov, and Michael E. Papka, ANL, papka@anl.gov
Novel Architectures: Volodymyr Kindratenko, University of Illinois, kindr@ncsa.uiuc.edu, and Anne Elster, Norwegian University of Science and Technology, anne.elster@gmail.com

Replicable Research Track: Lorena A. Barba, George Washington University, labarba@gwu.edu, and George K. Thiruvathukal, Loyola University Chicago, gkt@cs.luc.edu
Scientific Programming: Konrad Hinsen, CNRS, CRNS Orleans, konrad.hinsen@crnrs.fr, and Matthew Turk, NCSA, matthewturk@gmail.com
Software Engineering Track: Jeffrey Carver, University of Alabama, carver@cs.ua.edu, and Damian Rouson, Sercery Institute, damian@sereneryinstitute.org
The Last Word: Charles Day, cday@iop.org
Visualization Corner: Joao Comba, UFRGS, comba@inf.ufrgs.br

Abstract: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author’s or firm’s opinion. Inclusion in Computing in Science & Engineering does not necessarily constitute endorsement by IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length. IEEE prohibits discrimination, harassment, and bullying. For more information, https://www.ieee.org/about/governance/p19-26.html.

© 2020 IEEE. All rights reserved. Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated on the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Dr., Danvers, MA 01923 USA. Postmaster: Send undelivered copies and address changes to Computing in Science & Engineering, 445 Hoes Ln., Piscataway, NJ 08855 USA. Periodicals postage paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Corporation (Canadian distribution) publications mail agreement number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8 Canada. Printed in the USA.
Special Issue

4 Guest Editor’s Introduction
Computational Science and Engineering Education
Joan Adler

7 Preparing a Computationally Literate Workforce
Scott A. Lathrop, Katharine Cahill, Steven I. Gordon, Jennifer Houchins, Robert M. Panoff, and Aaron Weeden

17 Artificial Intelligence and Mobile Programming Courses for a Video Game Development Program in Chile
Nicolas A. Barriga and Felipe Besoaín

26 Engineers Code: Reusable Open Learning Modules for Engineering Computations
Lorena A. Barba

36 Research-Based Education on a Master of Science Degree in Professional Software Development
Cathryn Peoples

45 Computational Methods in Chemistry and Biochemistry Education: Visualization of Proteins
Polina Pine and Liudmila Ivanovna Paina

50 Hitting the Ground Running: Computational Physics Education to Prepare Students for Computational Physics Research
Amy Lisa Graves and Adam D. Light

62 Addressing the Cold-Start Problem Using Data Mining Techniques and Improving Recommender Systems by Cuckoo Algorithm: A Case Study of Facebook
Saman Forouzandeh, Atae Rezaei Aghdam, Soran Forouzandeh, and Shuxiang Xu

74 Multitier Stacked Ensemble Algorithm for Improving Classification Accuracy
Ramalingam Pari, Maheshwari Sandhya, and Sharmila Sankar

86 Software Engineering Strategies for High-Energy Physics Simulations Based on Quantum Field Theory
Tomasz Przedzinski, Maciej Malawski, and Zbigniew Was
DEPARTMENTS

99 Scientific Programming
Staged Computation: The Technique You Did Not Know You Were Using
Konrad Hinsen

Novel Architectures
High-Level Synthesis-Based Approach for Accelerating Scientific Codes on FPGAs
Ramshankar Venkatakishnan, Ashish Misra, and Volodymyr Kindratenko

104

110 Education
The HPC Certification Forum: Toward a Globally Acknowledged HPC Certification
Julian Kunkel, Weronika Filinger, Christian Meesters, and Anja Gerbes

Also in this Issue
Masthead
CS Information

Computing in Science & Engineering
Abstract—This special issue presents experience reports on computational education in various subjects of science and engineering. They include descriptions of early and current general courses at undergraduate and early graduate levels, as well as some computational exercises for specific courses. General computer science courses were intentionally excluded, because our emphasis is on applications of computers to science and engineering subjects.

OVERVIEW

Despite the enormous impact of computers in our lives, and their frequent use in the administrative side of education, most university course syllabi have barely changed over the years to reflect this. These days, most students have a required computing course, but it rarely is placed in context of their other courses or teaches relevant numerical algorithms for their field. Faculties around the world justify the lack of computational aspects in their courses with excuses like: it is too time-consuming and disruptive; we do not have funding; this is how it has always been. Such explanations could be due to lack of knowledge and therefore confidence on the part of the lecturers. Good faculty development and availability of support materials can help overcome this. It is also a good idea to bring the students’ knowledge into class content; e.g., by giving them a challenge in advance to answer a specific question.

This issue’s articles highlight courses in computational science and engineering, within traditional degree programs. A type of course covered in CISE before is the introduction to high-performance computing (HPC).\textsuperscript{1} Less emphasis is placed here on computer science courses, or degrees and certificates in computational science and engineering. Providing students with computing expertise in one or two courses is the challenge we choose to address here.

A 2006 special issue of CISE\textsuperscript{2} addressed Computational Physics education, either via the additional course or two, or an entire degree; one example of the latter is long-time CISE editor Rubin Landau’s article in that issue. The present issue has a more general coverage, but the context is similar.
PAST AND PRESENT
COMPUTATIONAL SCIENCE AND
ENGINEERING COURSES

This list cannot be all-inclusive and apologies
to authors of work not mentioned here. Several
well-funded initiatives exist aimed at course
development with integration of science and
engineering material with computational sup-
port, for example, a national initiative in Norway. ³

Computational physics courses going back to
the 1980 s have been described in the archival liter-
ature. Steve Koonin wrote an article about his
courses in several languages with the first in BASIC
dating from 1983 at Caltech.⁷ Peter Bocherds taught
a laboratory-format course in computational phys-
ics since about 1984,⁴ and CiSE Associate Editor-in-
Chief Steven Gottlieb taught his class since 1987.⁵ I
have been teaching a similar course since 1988,
which I described in a 2014 article.⁶ Although devel-
oped independently, these courses have similar
features, such as hands-on experiences in class.

Many such courses have been developed, in some
cases leading to the writing of a textbook. Notable
examples include: the early book in True BASIC by
Gould and Tobochnik,⁸ a 1997 tome by Giordano,⁹
and Joaquin Marro’s material in Spanish.¹⁰ In more
recent years, Titus Beu¹¹ published an excellent
book with Python and C/C++ code.

Many of the older courses and books have an
implicit orientation toward end-use of HPC, as
indeed do I. Today, new educational efforts
emphasize programming ease, relying on open-
source environments and tools such as
Jupyter,¹³ with diverse examples in science and
engineering; many of these have less of a physics
orientation than the texts mentioned above.

IN THIS ISSUE

The articles in this special issue cover a wide
range of subject areas. The issue had an open
call for papers, and was advertised in CiSE and
at several computational physics meetings, plus
via few personal letters from the guest editor.
The response was wider than expected and sev-
eral good manuscripts had to be declined.

One of the most exciting aspects for me is the
number of female authors, and another pleasing
observation is the range both of countries (USA,
Chile, Russia, and Northern Ireland), and of
fields, which ranged from engineering, physics,
chemistry, to video game development. Authors
came from academic and industrial research cen-
ters and a range of universities of different types.

Articles include “Preparing a Computationally
Literate Workforce” by Scott Lathrop, Katherine
Cahill, Steven Gordon, Jennifer Houchins, Robert
Panoff, and Aaron Weeden, from several universi-
ties and the Shodor Education Foundation. This
is a general article with an HPC orientation, and
discusses virtual shared courses. A more specific
topic is discussed in “Artificial Intelligence and
Mobile Programming Courses for a Video Game
Development Program in Chile,” by Nicolas A.
Barriga and Felipe Besoain from Chile, presenting
curriculum design for two third-year undergradu-
ate courses. Another undergraduate course for
noncomputer-science students is Lorena Barba’s
(CiSE Editor-in-Chief starting this year) “Engi-
neers Code: Reusable Open Learning Modules
for Engineering Computations,” which uses
Python and Jupyter and includes modules for
first-time programmers.

At the graduate level, there is Cathryn Peo-
lies from Northern Ireland’s “Research-Based
Education on a Master of Science Degree in Pro-
fessional Software Development.” Research-
based education describes the application of the
results of research to facilitate and complement
student learning.

Finally, there are two articles concerning natu-
ral science computational education. Polina Pine of
Loyola University in Chicago (based on a pedagogi-
cal method from Russia by Liudmila Ivanovna
Paina) wrote “Computational Methods in Chemis-
try and Biochemistry Education: Visualization of
Proteins.” This article gives an introduction to the
use of VMD with clear instructions to give students.
Amy Lisa Graves of Swarthmore College and Adam
Light of Colorado College present “Hitting the
Ground Running: Computational Physics Educa-
tion to Prepare Students for Computational Phys-
ics Research.” This article discusses which
computational skills might be best learned in the
curriculum (prior to research) versus during
research.

Please read and enjoy the special issue and
keep CiSE in mind for future articles about your
own courses and material. CiSE has an Education
Department to host such articles on an ongoing
basis.
ACKNOWLEDGMENTS

I wish to thank my mentors for support over the years, especially Amnon Aharony who organized the first Computational Physics course that I taught while still a postdoctorant. Peter Bocherds encouraged my involvement in International Computational Physics activities. More recently, discussions with some of the authors in this issue helped clarify my thoughts. I acknowledge support from the Technion Physics Department and the Technion CIS center.

REFERENCES


Joan Adler is a still active but recently retired Senior Research Associate with the Technion—IIT, Haifa, Israel. Her research interests include simulation and visualization of atomistic and electronic properties of materials and computational physics education. She received the Ph.D. degree in physics from the University of New South Wales, Sydney, NSW, Australia. Her students and postdoctoral fellows developed the AViz visualization code for atomistic and electronic systems. She is a Past President of the Israel Physical Society and recent Vice-Chair of the IUPAP C20 Commission. She is a member of the editorial boards of Computers in Physics and Computing in Science and Engineering. Contact her at phr76ja@technion.ac.il.
Preparing a Computationally Literate Workforce

Scott A. Lathrop
Shodor Education Foundation and the University of Illinois

Katharine Cahill
Ohio Supercomputer Center

Steven I. Gordon
The Ohio State University

Jennifer Houchins
Shodor Education Foundation and North Carolina State University

Robert M. Panoff
Shodor Education Foundation and Wofford College

Aaron Weeden
Shodor Education Foundation

Abstract—There is a saying, “Everything changes, but nothing changes.” We are realizing a rapid technological revolution in the development, deployment, and application of computing technologies within every discipline and every sector of society. Yet, our ability to respond to the well-documented need for a large, diverse, computationally literate workforce remains a challenge. We summarize our 35 years of lessons learned for preparing the workforce that can inform efforts to address this challenge. We have pursued a multiprong approach to reach instructors, researchers, professionals, and students on a national scale. Our efforts in scaling up and sustaining activities range from teaching computational thinking through imparting HPC skills. We have been able to scale up these activities through community efforts to share, cooperate, and collaborate. The potential for providing life-long learning to everyone wishing to expand their computational knowledge and skills is greater than any organization can achieve on its own.
and deliver top-notch workforce development programs to benefit all academic disciplines and all sectors of society. By building on the experiences of others, we can accelerate our collective efforts to prepare the workforce. *You do not need to, nor should you, “reinvent the wheel.”*

Our reference to the workforce includes those who are learning and using computing technologies. Our paper has focused primarily on high school students through graduates and postdocs. We include faculty teaching courses and those conducting research. And we include professional staff developing software and supporting information technology infrastructure needed by the community.

There is a saying, “Everything changes, but nothing changes.” We are realizing a rapid technological revolution in the development, deployment, and applications of computing technologies within every discipline and every sector of society. Yet, our ability to respond to the well documented critical need for a large and diverse high-performance computing (HPC) workforce remains an unmet challenge. Through building communities of practice, we can enhance the sharing of materials, lessons learned, and best practices to empower all programs to be more effective in preparing current and future generations of critically needed practitioners.

This article summarizes successful approaches for curriculum development and effective strategies for educating and motivating the preparation of an HPC workforce that can inform continuing efforts to address this challenge.

References to *science* are short-hand for all disciplines, and references to *computational science* include high-performance computing, high throughput computing, data analytics, and data science in all disciplines.

**EDUCATION GOALS**

People must understand the basic principles of modeling and simulation as well as the overall scientific goals of computer modeling before they focus on the programming and related technical aspects of modeling complex systems. That basic understanding can be achieved using a variety of tools from spreadsheets, to higher order languages, such as MATLAB, or with different programming languages. However, science and engineering models increasingly require the use of multiple processors operating in parallel. Thus, an understanding of parallel computing is also of critical importance.

There is a distinction, however, between education and training with respect to these activities, although the lines are blurred in practice. Using the example of parallel computing, it is important for people to be educated about parallel algorithmic approaches across a variety of scientific domains. Only in that context should they then be trained on parallel programming tools, such as OpenMP or MPI. The resulting parallel algorithms may serve us well for many generations, while the programming tools will evolve much more quickly.

We emphasize the *parallelism in nature* in order to understand the *nature of parallelism*. As students become more immersed in the science itself, they can better explore how the actual interactions in a model or simulation give rise to opportunities in the computational implementation of that model to leverage parallel architectures and algorithms. In so doing, they learn to map their problem onto the appropriate parallel approach given available computing resources. Understanding the science, understanding the range of dependent to independent aspects of the models in that science, and then implementing a parallel solution are primarily in the purview of *education*.

The technical aspects of the implementation, learning about compiler and profiling options, and learning how to use visualization software, can be accomplished by appropriate training. However, without the underlying educational foundation, students find themselves learning techniques without necessarily knowing why.

Another aspect of the educational mission is to help students understand scaling to make decisions based on “time to science” profiling and analysis of actual runs of a code.

A highly simplified formulation of the time to solution is depicted in Figure 1. For a given amount of work *W*, the time to solution in an idealized parallel world would decrease as *W/N* (the gray curve) where *N* is the number of processors. However, communication between processors to move the
data to be processed to the relevant compute nodes is typically proportional to the number of processors (or worse). This contribution to the time to solution can be approximated as $aN$ (the yellow curve), where $a$ is dependent on a whole host of factors that characterize the communication costs. As seen in Figure 1, the total time to solution is the sum of these two components, as represented by the blue curve, which shows that there is a maximum number of processors for a given workload that minimizes the time to solution. Understanding this reality, and understanding how it differs between different problems and model formulations, is best thought of as an education objective. One can be trained to do scaling experiments, but understanding that any given problem is going to have its own best-case utilization of processors and memory takes an understanding of machine architectures and algorithms.

Students pursuing computational science careers should learn to gain an appreciation for computational thinking to understand how computing can be used for scientific problem solving. They should learn parallel reasoning to reflect the inherent parallelism of nature. They should also learn quantitative reasoning since mathematics is used for solving many problems. People developing software need good software engineering skills including: developing reusable software; ensuring that scientific results are reproducible; verification that the software works; and validation to ensure that the science is correct.

**CURRICULUM DEVELOPMENT WITH INSTRUCTIONAL FACULTY**

Changes to the academic curriculum can have the most far reaching impact on the number of people learning computational science principles. However, instructional faculty interested in computational science often need guidance on how to integrate computational science materials into their classrooms given an already full curriculum.

We have conducted over 250 face-to-face workshops, visits to over 200 campuses and interactions with over 7000 undergraduate faculty and middle and high school teachers to focus on computing tools, exercises, and approaches that have been successfully incorporated into classroom settings and professional activities.

The approach Shodor\(^2\) and XSEDE\(^3\) have taken has been to demonstrate how to introduce modeling and simulation in existing courses across the curriculum, demonstrate the resources available (e.g., curriculum, modules, codes), while using PC/Mac based examples to get started with spreadsheets, system modeling, and agent modeling. Even though this is not at the HPC level of computing, this establishes a firm base and proven success for making the content of undergraduate math and science dynamic, visual, and interactive.

Campus visits are one or two day visits by a team of domain specialists, educators, and technologists to raise awareness of local, regional and national providers of computational and HPC resources and services, which can enhance educational and research benefits for the campus. The campus visits help to foster an increase in faculty interest to attend more extensive workshops in computational science and an increase in interest in faculty to sponsor internship opportunities for their students. A secondary benefit of visiting campuses is to interest students directly as they often attend the presentations and workshops that are being organized for their professors. Campus visits have resulted in the addition of minor programs\(^4\) in computational science that have institutional continuity beyond the interests of any individual faculty members. These programs consist of a series of courses aimed at introducing students to modeling and simulation, programming and algorithms, and a variety of applied computational science topics. Many are based on the undergraduate computational science competencies\(^4\) developed as a part of several National Science Foundation grants.
Over 35 years of running workshops, campus visits, and ongoing collaborations with faculty, we have observed that curriculum changes take time to implement. Many faculty require more than one workshop or other contact in order to fully understand which tools and examples are best suited for their courses. Even with that knowledge, it often takes three to five years for in-depth incorporation of the materials into a course. Faculty often take an incremental approach adding one or two exercises to their classes each time the class is taught, revising them as needed based on their experience with their students.

The key to producing real change is tied to building a community of faculty, both within and across disciplines, that can share their approaches and experiences. There are numerous examples of efforts that have been successful in establishing communities of instructional faculty interested in computational science. For example, the NSF funded Blue Waters, XSEDE, and Shodor Education Foundation conduct a range of introductory to more advanced workshops on computational science education.

As an offshoot of these workshops, such as the National Computational Science Institute, several discipline specific workshops and materials were developed, including 30 undergraduate petascale curriculum modules for multiple disciplines. The Computational Science Education Reference Desk (CSERD) and the HPC University portal were developed to facilitate searches for relevant materials and exercises by subject, keyword, audience, and other attributes. Workshop participants have access to a complete set of lecture materials and labs that can be completed using a variety of computational tools.

The most complete set of materials is in the area of computational chemistry. The Partnership for the Integration of Computation in Undergraduate Physics (PICUP) has created a set of computational physics materials and exercises, as well as a faculty forum to share computational physics exercises. There are a number of textbooks relating to computational science in general, physics, and numerical methods.

These efforts have resulted in significant progress in helping to integrate computational science into the curriculum. The data are incomplete as various projects have been inconsistent in following up with attendees. Nevertheless there is strong anecdotal evidence of success. The XSEDE project tracked the impacts of computational science education workshops over two years by soliciting responses by attendees. Those who responded indicated that over 100 courses were modified or created incorporating computational science modules. In addition, over 280 modules were reviewed by project staff and added to a digital library. In a survey of all past attendees from another set of workshops lead by the Shodor Education Foundation, over 50 faculty described how the workshops were translated into course changes at their institutions.

We have seen collaborative groups emerge to further advance the work we have begun. For example, PICUP continues to offer workshops and share materials with faculty on incorporating computational methods into the curriculum.

As computational science increases in importance, we expect the communities of faculty interested in each domain will expand and the materials will be further integrated into the curriculum. Shodor’s mailing list of the workshop alumni is used to provide information on relevant workshops, grants, and internships of interest to that community.

**MOTIVATING STUDENTS**

Classroom teaching is a first step for students to learn. However, we witness tremendous impact motivating and encouraging students through nonclassroom activities. Good faculty mentors are often a key to successful student engagement. Mentors make a big difference when they can identify students who have the potential to succeed and guide them along their way. Often this requires mentors to extend their reach to students who are not yet engaging in computational science or HPC. Even students who already have a basic understanding of what HPC is may
not yet envision themselves being a contributor to the HPC community. This seems to be the case for many students belonging to underrepresented groups who are aware of the lack of representation and support for people who belong to groups they identify with within the field. Embarking on an HPC career alone is a daunting task for a student, particularly one early in their education, particularly when the topics of HPC are not part of their standard curriculum. This is where an encouraging and supportive mentor can help fill the gap. Mentoring guidelines have been developed and used very effectively over the last ten years by the International HPC Summer School for the participating faculty, staff, peer mentors, and the mentees from Canada, Europe, Japan, and the United States.

A good mentoring relationship is enhanced when the mentor has a hands-on project to which the student can contribute. If the mentor is able to support the student not only over one summer but also in a subsequent academic year, the scope of the project can be expanded. This also allows for the mentor to guide the student through the process of preparing the work for publications and presentations in journals and at conferences and other events.

The benefit to students is enhanced by providing training and education to cadres of students who are working with project mentors. Pulling in students from diverse backgrounds and disciplines leads to a panoply of approaches to problem solving and helps each student feel part of a greater community. An in-person, hands-on workshop with engaging instructors can greatly increase students’ confidence with HPC and connectedness with others who are doing the same thing. Face-to-face workshops allow for personal connections that overcome the social limitations of online communication. Even the simple act of sharing a meal together can help foster connections that are much more difficult to establish in a purely online learning environment. We have modeled internship programs for over 200 students with this approach through Blue Waters and XSEDE.

We have been conducting effective workshops for students balance the sharing of insights by instructors with collaborative activities that help students develop an intuitive understanding of the architectures, applications, and algorithms. Teaching students how to recognize different paradigms in parallel computing helps them understand the applications on which they work with their mentors. Early, physical access to hardware helps students feel confident and see the value in what they are learning. This can be achieved whether the workshop includes a tour of an HPC center or utilizes small classroom-style clusters, such as LittleFe. Starting each lesson with a preview of the goals and the “why” of the lesson helps students engage with the activity and understand how it could relate to their own research. Having students give a presentation at the end of the workshop helps them reflect on what they learned and predict how it will impact their work moving forward. It also helps everyone see the breadth of research topics that are impacted from learning about HPC.

The way you teach is just as important as what you teach. In an in-person workshop, hands-on activities are often more favorable than lectures. Students are able to engage with, and more readily understand the purpose of, realistic scientific examples as opposed to simple “Hello, World” programs or application-independent discussions of algorithms. For example, using the application of heat transfer through a metal plate gives students an idea of the usefulness of Jacobi iteration, more so than simply introducing the algorithm by itself with no connection to an actual application. When working with code, having students modify an example that already works is favorable to creating an example from scratch or “filling in the blank” to complete a function or fix a bug.

It is also important to be cognizant of student attention in a workshop environment. We have found that it is important to teach today’s HPC students how to “wait productively” as many modern HPC environments require batch job submission as opposed to interactive in-situ
visualizations of running programs, which is a different paradigm than what many students are used to. This also applies to command-line interfaces, and while there are new tools being developed to help students avoid the command line for many cases of using HPC, learning to use a command line remains important for students to be confident working with a wide variety of systems.

Many students are now content with having all the lesson materials on their screen instead of handouts, but they often prefer to have everything in one view instead of switching between windows. Such considerations vary for each cadre of students and each individual learner, so it is important to work in flexibility and adaptability into the workshop curriculum and be willing to change things on the fly. These changes can be informed by surveys and other evaluation tools, which can help identify the impact of instructional decisions on student learning day-to-day during a workshop as well as long-term in the weeks, months, and years following the event. Such surveys along with longitudinal tracking of students’ career paths, publications, and presentations help demonstrate the long-term impact of instruction.

Workshops for students are enhanced by bringing in “near-peer” mentors who have been through the workshop themselves previously. These students can provide guidance and instruction from a recent participant’s perspective as well as refine their own skills by teaching others. This can add more confidence and expertise to a select group of students who have shown capability and interest in helping their peers. It also helps foster a spirit of giving back to the community.

TRAINING FOR LIFE-LONG LEARNING

To stay current, faculty, researchers, and professional staff benefit from opportunities to keep pace with advances in computing technologies that can accelerate their professional productivity and advance their careers. We have been conducting hands-on training with a focus on researchers from all domains of study and software developers since 1986 for thousands of participants in hundreds of events. These events have also been available to academic students who are unable to acquire this knowledge from courses taught within their institutions. Training has also been offered to administrators and IT staff. Participants have spanned academia, industry, and government agencies. Training individuals to be successful is a labor intensive and difficult process. There is no one-size-fits-all approach that has been perfected over the years. The variety of topics, individual backgrounds, and time available requires a complex mixture of efforts from multiday in-person workshops to short on-demand videos or tutorials to meet the needs of researchers working in very different disciplines. Major organizations with the means to make large investments in personnel and technology to support these efforts have come a long way with developing effective training in a variety of formats. These organizations mostly make these resources available to other institutions so effective training can be shared to the widest audience, but platform and environment differences can sometimes make this translation a complex one for smaller computing centers.

One-on-one consulting can be the most effective mechanism to educate an individual, but it is also very time and personnel intensive. In-person group training is the next most effective method for training people. The greatest challenge with in-person training of a group of people is that they tend to have very different backgrounds, levels of experience and understanding of the topics being presented, and goals. It is important to be clear about prerequisites, the topics to be covered, and the depth to be covered to better manage the expectations of the registrants - to be clear about what they will and will not learn. Including learning objectives is an added plus for potential registrants. An example of group training is the International HPC Summer School.

Many organizations (e.g., national HPC Centers, large campuses, and multisite organizations) serve
people who are not reasonably able to participate in person. There are effective methods for delivering quality training to remote participants via high-definition video conferencing, webcasts and self-paced tutorials, but these methods require more preparation and planning to be effective and of high quality. Simply broadcasting an in-person event is not an effective method of making this training available to a larger audience.

Improvements in high definition video-conferencing (HDVC) have allowed delivery of training in real time to multiple sites simultaneously. When using these technologies for groups of people gathered at remote sites, it is most effective to provide a local mentor to assist people with hands-on activities. Participants are able to ask questions of presenters at remote locations verbally and through social media tools such as YouTube and Slack. This approach is staff intensive as personnel need to be on-site to support the A/V infrastructure, to provide a knowledgeable mentor, and to coordinate the logistics of recruitment and registration of participants, as well as the handling of logistics of room and equipment reservations, and personnel commitments. As an example of this type of training offered and the effort to make the recordings of the sessions available, refer to the Petascale Computing Institute 2019.10

With the HDVC and webcast approaches, it is easy (such as with YouTube) to record the sessions (pending approval of the instructors) to make them available to participants to replay presentations or to view sessions they missed. The recordings can also be made available to larger audiences, or even made publicly available. As a word of caution, making recordings available requires compliance with the Americans with Disabilities Act to ensure everyone has full access to the content of the recordings. YouTube has capabilities for automatic transcription, though the transcription technology is not perfect.

The creation of over 100 self-paced tutorials,9 whereas very labor intensive to create quality materials, has benefitted the largest number of people. We have seen downloads of self-paced HPC related tutorials in the tens of thousands. Researchers benefit tremendously from this form of just-in-time learning. Further analysis is needed, but anecdotally people tend to view portions of these tutorials, rather than the complete tutorials. This may indicate people are looking for answers to specific questions.

Effective training includes well tested, hands-on activities to allow participants to practice what they learn. Design of informal training materials has benefitted in recent years from the outcome-based learning initiative methods. Better learning outcomes result from providing trainees with hands-on experience and opportunities to interact with the instructor.

Building Community

To implement effective education and training programs, it is important to raise awareness of resources, recruit participants, and provide access to quality learning resources.

Raising Awareness

It is important to make the workforce development offerings and materials available to those who may benefit. People do not know what they do not know and are largely unaware of the resources and services that are available. Researchers, faculty, and students frequently follow up on the information they learn through outreach (such as conferences, campus visits, and professional societies) to access and utilize the resources and services they learn about.

The XSEDE Campus Champions3 include over 600 individuals at more than 300 colleges, universities, and research-focused institutions, who help researchers become aware of and gain access to local and national computational resources and services. The Champions also assist each other and their organizations to improve services for their researchers, faculty, and students.

Professional societies provide opportunities for researchers as well as workforce development professionals to learn what their peers are accomplishing. These societies provide forums for professionals to exchange information they may utilize in their daily activities, as well as fostering collaborations on projects of common interest. For example, the ACM SIGHPC Education Chapter5 provides collaborative opportunities.
for addressing workforce development. The Chapter welcomes community contributions to expand the collective knowledge on the challenges and opportunities for enhancing HPC education. We encourage you to join the Chapter and attend Chapter organized webinars and conferences (e.g., SCxx, PEARCxx, and ISCxx) to remain informed and pursue collaborative projects.

Disseminating Learning Materials

The item of highest priority consistently requested by education and training professionals is access to a repository of quality materials. There are significant challenges we have learned from developing CSERD, the ACM SIGHPC Education Chapter, and HPC University.\(^2,5,4\)

Many organizations work to create their own repository (or list of resources) in order to serve their constituents, and thereby promote their own efforts. Unfortunately, people rarely take the time to submit materials to repositories external to their organization. As a result, there is no single repository that is commonly considered to be “the best source.” Repositories that attempt to be comprehensive are commonly left to search for and include materials on their own.

To help alleviate the “not invented here” syndrome, it would be beneficial to facilitate searching across sites by using common metadata tags for all entries. Repositories using common metadata can enable people to search across repositories. Repositories should routinely identify broken links to be corrected or removed; and avoid using databases that are inaccessible to search engines.

People are looking for repositories with materials that have been reviewed for their quality and their target audience. In-depth reviews are very difficult to achieve because people generally do not want to take the time to conduct a thorough review including verifying that the exercises work. People may at best provide a simple rating on a scale of 1–5, such as is done with Amazon. However, the HPC audience is relatively small, and we cannot expect many responses.

It is advantageous to provide a well-organized repository with materials categorized so that they are easy to search and browse. The XSEDE project provides a good model of categorizing training materials and providing roadmaps to assist people searching for training.\(^3\)

Delivering Virtual Shared Courses

To make HPC credit courses available to students at multiple campuses, Blue Waters, and XSEDE supported virtual shared courses\(^11,12\) by recruiting expert instructors to teach full-semester courses through recordings of their lectures for viewing in collaborating classrooms. Collaborating faculty at institutions across the nation were recruited, who were generally knowledgeable about the topic, but who did not feel they could teach the topic on their own. The collaborating faculty added the HPC course to their local course catalog, recruited students to enroll in the course within their own institution, mentored the students attending the course, and graded the students on their exams and a computational project.

Collaborating faculty have had positive experiences with a flipped classroom model of teaching in which the students watch the videos on their own time and participate in active discussions of the course content and hands-on activities during scheduled class time. Faculty generally felt that the flipped classroom approach is a better use of everyone’s time and allowed the students to go deeper into the course content.

A virtual shared course approach enables collaborating institutions to offer course topics that otherwise would not be possible, and allows collaborating faculty to gain confidence to subsequently teach the material on their own. The home institution retains the full-financial benefit of offering these courses and provides students with access to advanced HPC topics regardless of their institutional affiliation. The Big10 Academic Alliance CourseShare uses this approach among their member institutions. A sustainable “consortium” with a larger and more diverse mix of institutions is worthy of further discussion and exploration.
SUMMARY

Organizations should identify workforce development goals, objectives, constituencies, and metrics for success to serve with high-quality programs. All organizations are encouraged to share their lessons learned so that the community can “raise the bar” and effect large scale sustained computational science and HPC workforce development.

No organization has enough resources to address the broad range of HPC related workforce development needs. The community is encouraged to work together to tackle the many tough workforce development challenges including, but not limited to the following.

- Identifying and filling the education and training gaps from basic to advanced topics.
- Creating mechanisms for interconnecting repositories.
- Fostering sharing of virtual courses among institutions.
- Making learning materials accessible to people with disabilities.
- Translating learning materials into multiple languages.
- Keeping core competencies students should learn current.
- Advancing badging and certification processes.
- Scaling up quality programs to reach more people on a sustained basis.
- And updating this list as learning needs evolve.

Please share information about your own lessons learned, activities you are pursuing, materials you have to share, resources you have found to be useful, and your collaboration ideas.

REFERENCES


Scott Lathrop is currently a Blue Water’s technical program manager for Education with Shodor Education Foundation, Inc., Durham, NC, USA. He has been involved in high-performance computing and communications activities since 1986, with a focus on HPC education and training for more than 20 years. He coordinates the education, outreach, and training activities for the Blue Waters project. He helps ensure that Blue Waters education, outreach, and training activities are meeting the needs of the community. He has been also involved in the SC Conference series since 1989, was a member of the SC Steering Committee for six years, and was the Conference Chair for the SC’11 and XSEDE14 Conferences. He launched the Campus Champions program during the NSF funded TeraGrid project. He formed and led the International HPC Training Consortium for three years, after which it merged within the ACM SIGHPC Education Chapter during the SC17 Conference. He has been active in the planning and participation in HPC education and training workshops at numerous conferences including the ISC, PEARC, and SC Conferences. He is the corresponding author of this article. Contact him at lathrop@illinois.edu.
Katharine Cahill is the education and training specialist with the Ohio Supercomputer Center (OSC). She has used computational modeling in the context of chemistry and currently develops training programs for both local and national high-performance computing resources. She is also the Education lead for XSEDE at OSC, where she aids in computational science curriculum development through faculty training. Prior to joining OSC, she was a computational chemistry postdoctoral researcher with Ohio State University investigating protein function and interaction with organophosphorus nerve agents. She received the B. A. degree in classics/classical languages, literature, and linguistics from Reed College, Portland, OR, USA, and the Ph.D. degree in organic chemistry from the University of New Hampshire, Durham, New Hampshire. She is a member of the Association for Computing Machinery and a board member of Education Chapter of the Association for Computing Machinery’s Special Interest Group on high-performance computing. Contact her at kcahill@osc.edu.

Steven Gordon is a professor emeritus with City and Regional Planning, Ohio State University, Columbus, OH, USA, and was formerly the senior education specialist with the Ohio Supercomputer Center (OSC). He was also the interim executive director of OSC for three years and the senior director of Education and Client Services. He led the education program of the Extreme Science and Engineering Discovery Environment program from 2011 to 2016, and managed the Blue Waters Graduate Fellowship Program. He was the founding chair of the SIGHPCC Education chapter and currently serves as an at-large member of the board. Contact him at sgordon20@gmail.com.

Jennifer Houchins is a doctoral student with the College of Education, North Carolina State University, Raleigh, NC, USA, with a program area focus of learning, design, and technology in the Department of Teacher Education and Learning Sciences. Her research interests include instructional design and the use of computing in education, particularly as it relates to computational science and computational thinking practices across disciplines. She received the Bachelors of Science degree in mathematics from East Tennessee State University and the Master of Science degree in mathematical sciences from Clemson University. She is a member of the Association for Computing Machinery, the Association for the Advancement of Computing in Education, and the Association for Educational Communications and Technology. Contact her at jhouchins@shodor.org.

Robert M. Panoff is founder and executive director of Shodor, a nonprofit education and research corporation in Durham, NC, USA, dedicated to reform and improvement of mathematics and science education through appropriate computational and communication technologies. As PI on several National Science Foundation and US Department of Education grants that explore interactions between technology and education, he develops interactive simulation modules that combine standards, curriculum, supercomputing resources, and desktop computers. In recognition of his efforts in college faculty enhancement and curriculum development, Shodor was named as a NSF Foundation Partner for the revitalization of undergraduate education. In 1998, Shodor established the Shodor Computational Science Institute, which was expanded with NSF funding in 2001 to become the National Computational Science Institute. Shodor’s Computational Science Education Reference Desk serves more than 4 million webviews per month as a Pathway portal of the National Science Digital Library. He consults at several national laboratories and is a frequent presenter at NSF workshops on visualization, supercomputing, and networking. He received the M.A. and Ph.D. degrees in theoretical physics from Washington University, St. Louis, MO, USA, with both pre- and postdoctoral work with the Courant Institute of Mathematical Sciences at New York University. He received an honorary Doctor of Science degree from Wofford College in 2005 in recognition of his leadership in computational science education; he has been the recipient of the 2014 SIGCSE Outstanding Contribution to Computer Science Education Award. Since 2016, he has been a scientist-in-residence and visiting professor with Wofford College. Contact him at rpanoff@shodor.org.

Aaron Weeden is a parallel computing mentor with the Shodor Education Foundation, Inc., Durham, NC, USA. His research interests are in the areas of computational science education and parallel computing education, especially at the undergraduate and precollege levels. He received the Bachelors of Arts degree in computer science from Earlham College. He is a member of the Education Chapter of the Association for Computing Machinery’s Special Interest Group on High-Performance Computing. Contact him at aweeden@shodor.org.
Artificial Intelligence and Mobile Programming Courses for a Video Game Development Program in Chile

Nicolas A. Barriga and Felipe Besoaín
Universidad de Talca

Abstract—In this article, we present our curriculum design process for two-third-year undergraduate courses: artificial intelligence for video games and mobile device programming. These are part of a 4.5 year program in video game development and virtual reality engineering. We explore the range of possible content, usually aimed at traditional computer science or software engineering students or practitioners, and extract and adapt it to our particular program. Students developed high quality apps, achieved good standings in AI competitions, and some even published peer-reviewed articles. We believe the approach presented in this article is broadly applicable, but can be especially useful to instructors creating courses for uncommon and innovative programs.

In recent years, several attempts have been made at using video games to teach general artificial intelligence courses. However, there has not been much discussion on how to teach Video Game AI. Video game AI techniques, as used in industry, have some overlap with standard academic AI topics, such as best-first search. Still, traditional AI courses rarely teach the basic methods of video game AI, such as: decision making via finite-state machines (FSMs), behavior...
trees (BTs) or utility; hierarchical pathfinding; or procedural content generation (PCG).

The spread of information and communication technology (ICT) in the population has increasingly grown, in particular mobile phone technology. Smartphone technology continues to be the principal motor of growth for ICT. According to global mobile market report (newzoo.com), smartphone penetration in developed countries is around 70% to 80%, while in developing countries it is roughly between 25% and 50%, and growing steadily. Mobile devices are a key market for video games accounting for 45% of the USD$152Bn global video game revenue, and 76% of the USD$92Bn global mobile apps revenue. In this context, knowing how to develop mobile applications is an important skill to develop during this undergraduate program.

This work describes our approach for creating and teaching two undergraduate courses. 1) Artificial intelligence for video games; and 2) mobile device programming. The curriculum has been designed in a way that the second course reinforces the first, through projects and practical applications, where the students apply all the theoretical algorithms, learned in the first course, in the development of mobile applications. It draws inspiration from both academic and industry literature, as well as from direct communication with game AI professionals.

PROGRAM DESCRIPTION

The Video Game Development and Virtual Reality Engineering undergraduate program is aimed at students who want to develop their creativity and technological skills with applications in virtual environments.

The program presents a multidisciplinary graduate profile with the aim of providing students with the competencies needed for the creation of video games with an emphasis on informatics, software development, and the application of information and communications technology to this area. The curriculum blends the fields of engineering and design to instruct professionals with scientific and creative competencies for working in the interactive entertainment and virtual simulation industry, developing entrepreneurship, and addressing diverse areas such as recreation, simulation, education, and training, among others. The graduates will have a solid foundation in basic design and computer sciences, with a modern, proactive focus.

The graduate’s distinctive characteristic will be reflected in his/her entrepreneurship and leadership skills for executing innovative solutions using information technologies to develop videogames and applications, incorporating teamwork skills. Moreover, the graduate’s multidisciplinary training will allow him/her to contribute in the development and management of creative industry and software projects, as well as to take on emerging technologies and new areas.

COURSE AUDIENCES AND SCOPE

The curriculum of the Video Game Development and Virtual Reality Engineering undergraduate program is 4.5 years long, divided into three blocks. First, two years of basic science and foundation courses of each discipline (design, computer science, general education, foreign Language), leading to an associate’s degree. Second, two years of specialized and detailed instruction, complementing the first block, leading to a bachelor’s degree. Finally, with the final degree project, skills related to innovation and entrepreneurship are stressed, and a first professional degree is obtained. The program summary shows prerequisites for the courses described in this article.

Early in the conception of the Video Game Development and Virtual Reality Engineering program we decided that harmonization between courses would play a large role in the design process. We coordinated several small clusters of courses, such as the one presented in this article, as well as larger groups. Each block includes a workshop course with a high credit load and 7 to 8 h in a laboratory, working in a project that ties all the competencies learned in the previous courses (see courses in yellow given by Besoain). For example, the first of these workshops (2D Game Programming Workshop) caps and integrates the knowledge, skills and competencies acquired in three product design courses, three programming courses and three video game development courses. This is also
considered a checkpoint where professors can check if the students are developing the competencies and skills properly or if they need some support on their progress.

Students are also exposed to other early experiences of course articulation, such as making one game project for three courses in the same semester, as seen on the courses in green given by Besoain. In this case, each course contributes with their disciplinary competency: programming, concept design, and video games basics.

Taking these workshop and introductory courses as an example, we designed a coordinated process between Artificial Intelligence for Video Games and Mobile Device Programming courses, aimed at offering students the chance to apply and develop their competencies in a related area, going from theory to practice.

Artificial Intelligence for Video Games is a one semester, 6 European Credit Transfer and Accumulation System (ECTS) credits (1 credit is around 25 to 30 h) long course. Our students are in the fifth semester of the 4.5 years Video Game Development and Virtual Reality Engineering undergraduate program. They have already taken several computer science courses, such as structured programming, object-oriented programming, databases, algorithms, and data structures, as well as various video game development courses and workshops. Mobile Device Programming, a 4 ECTS credits course, is delivered the following semester.

Our aim is to cover AI techniques currently in use in video games (e.g., FSMs, BTs, A*, constructive PCG), newer methods that are slowly finding their way into commercial games (e.g., Monte-Carlo Tree Search, MCTS) and novel techniques that might be useful in the close future (e.g., machine learning and search-based PCG). Due to the expanse of the subject matter at hand, we cannot cover every topic in detail. We will use a mix of guided programming activities (labs), solo programming assignments, research assignments, readings, and lectures. The content deemed most relevant to the students’ future careers will be taught in depth, using all four methods, while topics whose application to video games is more speculative will be given less attention (e.g., by only assigning a reading).

On the other hand, in the mobile devices course, we aim to study the growth of this technology and its applications for programming mobile applications. In this context, we work with Android OS due to its open-source license and SDK available to the developer community. The content goes from building a strong foundation in the Kotlin language (e.g., by resolving problems without a GUI, using only the core language), then understanding the Android OS and applications architecture, to programming apps that require access to sensors like accelerometers, GPS, light, etc., and services such as databases, Firebase among others. With this in mind, students build a portfolio of developed apps with different functionalities. They will later develop a final course project.

In the case of the two courses presented here, we have decided to coordinate the more theoretic artificial intelligence and the largely practical mobile devices courses by having the programming assignments in the latter use AI algorithms. We have settled on single- and two-player board games in a mobile platform, which allows us to include single-agent and/or adversarial search algorithms.

CONTENT

In this section, we present a compilation of the most relevant content we considered, as well as the content we finally selected for each course.

Artificial Intelligence for Video Games

We consider content covered by game AI books targeted at industry professionals, books covering academic game AI research, as well as some discussions in the AI Game Programmers Guild mailing list. For details, see Table 1.

Millington and Funge’s Artificial Intelligence for Games is the de facto standard textbook for game AI professionals. It covers a wide range of topics in great detail, while focusing on the methods themselves, rather than on the applications.

Mat Buckland’s Programming Game AI by Example is much more application oriented. It starts from example games, and pulls in the necessary knowledge, leading to a more applied, but
A recent book coming from academia is Yannakakis and Togelius’ *Artificial Intelligence and Games*. Here, the focus is clearly on the methods, in particular newer methods coming from academia, most of which have not been used in commercial games. It is written as a survey of the subject, giving a broad overview, but glossing over the implementation details. It contains numerous references for those wanting to dig deeper.

We have not covered some closely related, but very subject specific books, such as Dave Mark’s *Behavioral Mathematics for Game AI* on utility-based AI, Brian Schwab’s *AI Game Engine Programming* and Shaker et al.’s *Procedural Content Generation in Games*.

Finally, we compiled our own personal communications with professional video game programmers, as well as discussions that have taken place in the AI Game Programmers Guild, a forum for industry professionals. Most developers agree on the broad topics of pathfinding, decision making, and PCG, with some differences in specific algorithms. For example, A* and FSMs are seen as old—by some—and superseded by hierarchical pathfinding and BTs, while others prefer to include them, to build up from the basics. Adoption of other topics is more fragmented: movement and animation are seen as important, but they are usually covered in other courses in game programming programs; adversarial search and machine learning are considered niche areas, not part of the regular AI programmer’s toolbox, but useful tools if you have them.

**Selected AI Content**

In this section, we present the topics we have chosen to cover. We dismissed topics already seen by our students, such as movement and animation. We start with a review of basic search algorithms, since a working knowledge of them is crucial to later subjects. We then move to pathfinding and decision making, of which there is an agreement both in the literature and in the practitioner community. We then spend a significant amount of time on tree search algorithms. We believe that they have a big niche of current applications in board games, as well as potential new uses in the near future in card and strategy games. We finish the semester with an overview of PCG, both using traditional methods and emerging AI-based ones. A summary can be viewed in Table 2.

We believe this selection of content provides our students with a good balance of job-ready skills and a broad algorithmic understanding of the field.

**AI Content Delivery**

The AI content was delivered using a flipped classroom approach.
weekly readings—mostly chapters from the Game AI Pro\textsuperscript{11} and AI Game Programming Wisdom\textsuperscript{12} book series—followed by short questionnaires at the beginning of each class to assess students’ understanding of the material. Then, the instructor would fill in any perceived gaps in knowledge and follow with group exercises. Lab time was spent, on alternating weeks, on guided programming tasks, or individual programming assignments.

Mobile Device Programming

Similar to the AI for video games course, for mobile device programming, we considered content covered by several Android development books targeted at academic and industry professionals, and also the official Android development website.

Dawn and David Griffiths’ *Head First Android Development: A Brain-Friendly Guide*,\textsuperscript{13} is particularly interesting because of their approach in the *Head First* series that is based on cognitive science and learning theory, using a visually rich format to engage the reader rather than a text-heavy approach. This approach proves beneficial to engaging students and enhancing their learning process, both inside and outside of the classroom.

Neil Smyth’s *Android Studio 3.0 and 3.5 Development Essentials: Kotlin Edition*\textsuperscript{14,15} is a complete book that presents Android Studio and does the walkthrough to set up an Android development and testing environment. It continues with an introduction to programming in Kotlin, including data types, flow control, functions, lambdas, and object-oriented programming.

Ian F. Darwin’s *Android Cookbook: Problems and Solutions for Android Developers*,\textsuperscript{16} introduces work with accelerometers and other Android sensors, the use of various gaming and animation frameworks, the storage and retrieval of persistent data in files and embedded databases, and the access of RESTful web services with JSON and other formats.

Android has had a considerable growth rate, from the very first versions to the last one, improving their API and frameworks, and also changing their main programming language from JAVA to Kotlin. Therefore, as a technical resource, the official Android development website is in fact the best source of up-to-date examples and descriptions of each concept, process, or SDK related topic.

Selected Mobile Device Content

In this section, we present the topics we have chosen to cover. We dismissed topics already seen by our students, such as the object-oriented programming paradigm and UI design.

The course is planned as follows, with three main units: 1) Kotlin language essentials, which cover the technical aspect of the language with several analyses of different apps, smart context, and ubiquitous computing topics. 2) We start understanding the architecture of Android OS and its apps, activity lifecycle, and an incremental exploration from GUI, events, states, access to sensors, and external services. 3) The main approach in this unit is to explore a solution to a

<table>
<thead>
<tr>
<th>Topic</th>
<th>Weeks</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>2</td>
<td>Trees.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Graphs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Breadth-first search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depth-first search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dijkstra.</td>
</tr>
<tr>
<td>Pathfinding</td>
<td>3</td>
<td>Map representations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JPS+.</td>
</tr>
<tr>
<td>Decision</td>
<td>4</td>
<td>Decision trees.</td>
</tr>
<tr>
<td>making</td>
<td></td>
<td>Finite-state machines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Behavior trees.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Utility.</td>
</tr>
<tr>
<td>Tree Search</td>
<td>3</td>
<td>Game tree search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Search optimizations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Evaluation functions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monte-Carlo tree search.</td>
</tr>
<tr>
<td>PCG</td>
<td>2</td>
<td>Traditional methods: Constructive, fractals, noise, grammars.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AI methods: Search-based, ML.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Video game AI course topics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Search</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Pathfinding</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Decision making</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Tree Search</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PCG</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
problem proposed by the student, or selected from instructor suggested topics, that can be addressed with a mobile app. We expect to complement and integrate all the previous competencies and skills. A summary is presented in Table 3.

**Unit 1** This unit introduces the essentials to the new programming language, with the aim of solving technical problems. We finish with a project where students have to implement a game that requires an AI algorithm. These algorithms where previously taught, both theoretically and practically, and the goal here is to have a first approach to solving problems in console with Kotlin.

**Unit 2** Since the technical aspects of Kotlin programming are covered, we start integrating mobile device specific aspects. We create different apps, incrementally adding new topics, with the aim of introducing a general approach to application development using different resources. Here, a second integration comes with the implementation of the first project, now in a mobile environment, where they need to take care of the GUI design, events, and states of activity, among other important aspects.

**Unit 3** The last unit aims to serve as an integration arena for solving a problem that requires the use of external services. It allows students to refine their previously acquired knowledge while encouraging their autonomy.

The integration of knowledge and competencies is very important for courses to provide the basic skills for the creation of products that require a multidisciplinary point of view. Even though mobile apps can be developed by a single developer, professional development considers many aspects besides programming. Larger teams manage by pooling together people with different skill sets. Our program stresses this multidisciplinary aspect, by having students undertake user experience (UX) and design aspects as well as technical and algorithmic aspects.

**RESULTS**

In the AI course, a programming assignment was linked to external AI competitions at the IEEE Conference on Games (IEEE CoG). One of the groups placed in fifth position, out of 12 participants, in the μRTS competition. It is worth noting that their entry beat the previous year’s winner, and that all higher placed entries were submitted by graduate students.

The same course had a research assignment, where students were asked to write a report on a topic of their choosing, related to video game AI. Two of those reports were later turned into exploratory surveys and accepted for presentation at the IEEE Chilecon 2019 conference. A third one is currently under review at a journal.

In the *Mobile Device Programming* course, students go on to develop several applications. In the first unit, they develop Kotlin console apps. Figure 1 shows a Microrobots puzzle being solved using a single agent search or

<table>
<thead>
<tr>
<th>Table 3. Mobile device programming topics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Unit 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Unit 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Unit 3</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
pathfinding algorithm from the AI course. The game consists on finding a path from a starting position to a destination, moving on cardinal directions, between squares with the same number and/or color.

On the second unit, they developed a full-fledged game application, Connect4, shown in Figure 2, which requires an adversarial tree search algorithm.

**DISCUSSION**

These syllabi are specifically tailored to our students. For students with a greater algorithmic knowledge, the first section of the AI course, on search can be skipped. Conversely, students new to game programming will probably benefit from a section on movement and animation.

If the program curriculum included a second AI course, we would move PCG and tree search there. That second course would also contain machine learning and reinforcement learning modules, as well as player modeling. The extra time in the first course would be used for expanding our coverage of pathfinding algorithms, specifically world representations, and adding single-agent planning algorithms to our decision-making module.

Likewise, the mobile device programming course is also tailored to our students. The two main considerations were: 1) algorithmic knowledge and previous programming courses (structured and object-oriented OO); and 2) courses in the same semester, which are: *Operating Systems and Networking* and *Usability and Interfaces*.

The first unit focuses on Kotlin’s essentials, considering our students already know the OO paradigm and JAVA language. If the first OO programming course was taught in Kotlin, we could probably skip this unit.

The second unit, which focuses on Android essentials, can be extended depending on the goals of the program. In this case, we aim for giving the students a strong basic foundation that allows them to keep learning and improving their skills and competencies autonomously. It would be very interesting to include topics such as Media resources, OpenStreetMap (OSM) and Google maps (in detail), Android instant apps, Google Play store publishing, or cloud storage, among others. But, by giving students the freedom to choose their own project topics, we allow them to explore some of these topics or just deepen their knowledge of previous ones.

Finally, it is worth noting that the courses presented here are a good complement to our department’s main research fields: ubiquitous and pervasive computing, mHealth, uHealth, gamification, UX, video game development, and video game AI. Even more to the point, these courses are a very good match for the authors’ research topics, as evidenced by their latest papers on gamification of a mobile app for HIV prevention and on using supervised learning, search algorithms, and reinforcement learning for video game AI.

**CONCLUSIONS**

Our program covers the major sources of game AI content by professional practitioners along with academic researchers in the field.
Where we found significant overlap, we added it to our syllabus, with some exceptions where our students had already covered the topics in previous courses. On subjects in which there was some disagreement, we have opted for content we expect will become more relevant to AI practitioners in the video game industry in the near future. We have also made suggestions on how to adapt this course to your own program curriculum, by taking into account previous courses, or the possibility of more than one game AI course.

In the same way, for mobile devices, the course allows the students to create a strong foundation for developing mobile apps. But it is important to take into consideration the convergence of knowledge of all the previous and parallel courses, which allow explaining easily some important parts of the technical point of view of programming. Having as a tool the integration of previous topics (i.e., AI algorithms), allows the students to develop projects to familiarize themselves with a new language, while separating the technical challenges, from the algorithmic ones. Conversely, the earlier course can establish strong theoretical foundations without the need to develop full-fledged applications. Finally, the students learn to iterate the same solutions, from algorithms, to basic implementations, to mobile applications, making mental connections from the computer science, software engineering, and product design aspects of application development.

We have found that the connection between subsequent courses is well received and culminates in course projects that are more advanced and compelling for both students and instructors. The course design presented in this article has remained stable for the last two years. Due to its success, we plan on integrating further AI algorithms into the mobile device programming course for next year’s offering.

**REFERENCES**

Nicolas A. Barriga is currently a Professor with the School of Video Game Development and Virtual Reality Engineering, Faculty of Engineering, Universidad de Talca, Talca, Chile. He received the Ph.D. degree with the University of Alberta, Edmonton, AB, Canada, for his work on state and action abstraction mechanisms for RTS games. His current research interests are in the broad area of video game AI. Contact him at nbarriga@utalca.cl.

Felipe Besoain is currently a Professor with the Bioinformatics Department, School of Video Game Development and Virtual Reality Engineering, Faculty of Engineering, Universidad de Talca, Talca, Chile. He received the Ph.D. degree with the Universitat Oberta de Catalunya, Barcelona, Spain, for his work in mobile devices and ubiquitous computing. His research interests are the application of information technologies in intelligent contexts for bioinformatics, agronomy, and health areas; and the use of ubiquitous computing and mobile devices, gamification, and virtual reality applied to mHealth. Contact him at fbesoain@utalca.cl.
Engineers Code: Reusable Open Learning Modules for Engineering Computations

Lorena A. Barba
Mechanical and Aerospace Engineering, The George Washington University

Abstract—Undergraduate programs in science and engineering include at least one course in basic programming, but seldom presented in a contextualized format, where computing is a tool for thinking and learning in the discipline. We have created a series of learning modules to embed computing in engineering education, and share this content under permissive public licenses. The modules are created as a set of lessons using Jupyter notebooks, and complemented by online courses in the Open edX platform, using new integrations we developed. Learning sequences in the online course pull content dynamically from public Jupyter notebooks and assessments are autograded on-the-fly, using our Jupyter Viewer and Jupyter Grader third-party extensions for Open edX. The learning content is modularized and designed for reuse in various formats. In one of these formats—short but intense workshops—our university library is leveraging the curriculum to offer extracurricular training for all, at high demands.
basics. Less common—even though it is known to be effective—is to teach programming in context. The classic example is the media-computing course introduced nearly two decades ago at the Georgia Institute of Technology (for liberal arts, architecture, and management/business majors).\textsuperscript{1,2} Evaluation efforts on that multiyear curricular innovation support the idea that context-based teaching of programming increases student motivation and success.\textsuperscript{3,4} (In the Georgia Tech experience, it also reduced the success gender gap.) A more recent effort to introduce contextualized computing education in engineering found that it was effective in enabling students to apply computational practices to continue learning in their discipline.\textsuperscript{5} In view of their observations, the authors recommended integrating context-based computing early and often in the engineering curriculum.

In this article, I describe an initiative to develop a series of learning modules aimed at integrating computing in the undergraduate engineering curriculum. The modules adopt the context-based format for teaching programming, and are also designed to be reusable, and shared under standard public licenses (CC-BY for content and BSD-3 for code). We first developed three learning modules—each adding up to about one university credit of work—and taught a second-year engineering course based on this content in Fall 2017 and Fall 2018. The faculty of the Mechanical and Aerospace Engineering department then approved creating a two-course series in computing; we used the first two modules for a first-year course in Spring 2019, and started writing two additional learning modules to complete a revised second-year course. We are working with colleagues to develop additional learning modules to use within core engineering courses, aiming to reinforce the ability of students to use computational practices and problem-solving in their discipline. The design patterns we adopted (described in the next section) are the product of many years of teaching computational topics to engineering students, and adopting increasingly popular technologies (especially Jupyter notebooks). We use a hybrid-learning approach, combining an online course platform (Open edX) and autograding of student assignments, and active learning in the classroom meetings. This project fully embraces open-source tools and open educational practices, and its goal is to advance innovation in engineering education by integrating computing across the curriculum, disseminating our products broadly, and inviting collaboration.

KEY CONCEPTS AND DESIGN PRINCIPLES
Some key concepts and design principles in the \textit{Engineers Code} series of learning modules are:

1) \textit{The idea of “computable content”:} educational content made powerfully interactive via compute engines in the learning platform;
2) \textit{The idea of open pedagogy:} reflecting in the teaching practice the ethos and practices of open source software;
3) \textit{Modularization:} creating stackable learning modules that break-up the standard “course” format;
4) \textit{Harnessing the worked-example effect:} empirically shown to be superior to problem-solving for novice learners;
5) Using live-coding to structure active-learning class experiences; and
6) Guiding learners to document their own work.

Our chosen learning platform is Jupyter: a browser-based interactive computing environment, concretized in a document format that seamlessly interleaves code with text-based and multimedia content: the Jupyter Notebook. Each learning module consists of four or five fully narrated Jupyter notebooks (the lessons), and student assignments also prepared and submitted as Jupyter notebooks. I started using Jupyter for teaching in 2013 (before it adopted this name). Based on a practical module used in the classroom in my computational fluid dynamics (CFD) course (taught from 2010 to 2013 at Boston University), my first series of fully narrated notebooks is “CFD Python: The 12 steps to Navier-Stokes Equations.”\textsuperscript{6} Based on the experience creating and using the CFD Python learning module, and following a similar approach in later courses, we adopted this basic design pattern for developing lessons using computable content.

1) Break it down into small steps.
2) Chunk small steps into bigger steps.
3) Add narrative and connect.
4) Link out to documentation.
5) Interleave easy exercises.
6) Spice with challenge questions/tasks.
7) Publish openly online.

The Engineers Code learning modules are published as open educational resources (OER): anyone can access, reuse, revise, and redistribute the materials. The idea of creating educational materials that are made to be reused goes back 25 years, and led to the development of content licenses like creative commons. Recurring topics in the conversations around OER are the high cost of textbooks, increasing access to content (for worldwide learners), questions of copyright and licenses, and values around altruism and the public good. But arguably OER have not been transformational: various surveys show that faculty for the most part still require that students purchase commercial textbooks, and have little awareness of OER alternatives. Even if the open education movement was inspired by open source software, it missed some key features: open development, networked collaboration, community, and a value-based framework. In open-source development, we cherish our productive freedom: the freedom to work and collaborate by our own conventions, sidestep the restrictions of copyright law by attaching a license to our products, and prioritize access, distribution, and collaboration. In OER development, the narrative is often about creation of content, and adoption by others. There is the Author, and there is the Adopter, or User. Although Creative-Commons licenses are meant for reuse and remix, in practice, the emphasis is on sharing for reuse “as is.” (Consider for example the MIT Open Course Ware initiative: faculty create their course materials and deposit them for free access; users cannot become contributors.) Deliberately embracing the ethos and practice of open source software may not only lead to greater reuse, but could inspire students to more collaboration.

Modularization of the content is also aimed at increasing reuse. Course instructors all know how difficult it is to adopt another instructor’s materials to teach a course, as we almost always would teach it differently. Science and engineering textbooks are often excessively long, and courses based on them leave out substantial portions. Each instructor may choose different chapters to leave out, while all students still must buy the bloated full book. Unbundling a university course into smaller content units empowers instructors to “mix-and-match” and adapt their course to local goals and student cohorts. It also reflects better how learners consume content today, and how users interact with all sorts of media on digital platforms. Modular content design also enables just-in-time use, whether in support of a longer course, or through informal learning opportunities (e.g., short tutorials, library workshops, peer learning groups).

In the Engineers Code series, each module is designed to comprise about one credit-hour of course work, and take about four or five weeks to complete in regular course scheduling. Each module, in turn, consists of a handful of lessons written as Jupyter notebooks. And each lesson develops a topic through a scaffolded design that takes advantage of the worked-example effect: one of the best known and most widely studied of the cognitive load effects. The effect is positive when providing full guidance on how to solve a problem results in better student performance than problem-solving conditions with no guidance. Many studies have shown significant learning improvements with worked examples, compared with free-form problem-solving, for novices learning complex topics. The opposite is called generation effect: when learners generating responses perform better than learners in a presentation setting that provides an answer. Chen et al. concluded that the “worked example effect occurs for complex, high-element interactivity materials that impose a heavy working memory load whereas the generation effect is applicable for low-element interactivity materials.” Thus, strong guidance is best for novice learners dealing with complex materials; as students become more expert in the subject, guidance can be reduced in favor of discovery-based learning. Needless to say, learning to program imposes a high cognitive load, suggesting that inexperienced trainees will benefit from worked-example effects. Each lesson in our collection aims to harness this effect by breaking down a computational problem into atomic steps, providing a detailed narrative and documentation of those steps, and peppering the narrative with low-stakes exercises for students.

In the classroom, we adopt the method referred to in open source communities as
live coding. The instructor projects an interactive computing session (a draft Jupyter notebook), and demonstrates live the problem-solving sequence by typing and executing every command, while students follow along in their own interactive session. Naturally, mistakes happen, and the instructor has the opportunity to vocalize the corrections to these mistakes, modeling a process (debugging) that novices can be frustrated by when working alone. Students following along in a class can mistype, or decide to try a small change, and encounter different errors during class. To help them, the instructor is aided by in-class learning assistants: undergraduate students who have recently completed the course. We use the familiar Software Carpentry “Post-It Notes” method: all students are handed one pink and one green sticky note. When they have a problem or a question, they stick the pink note on their computer monitor (and a learning assistant can come to them); when they are finished with a class exercise, they stick the green note on their monitor (and the instructor can get a feel for the room with one look around). The live-coding approach, supported by in-class learning assistants, leads to an active learning format that works. Active learning is known to be superior to lectures both in terms of comprehension and recollection (memory).

Module 1: Get Data Off the Ground With Python

The first module assumes no prior coding experience, so the first three lessons are focused on creating a foundation with Python programming constructs, with minimal and basic mathematical content (to spare cognitive load). The fourth lesson introduces the basic data structure in scientific computing: arrays; the mathematics content is kept at high-school level. The final lesson is a worked example of linear regression with real data, and contains some calculus-based content.


Lesson 3: Strings and lists in action—A full example using what you learned in lessons 1 and 2: playing with a text file containing the MAE Bulletin (list of courses with their numbers, description, prerequisites). Reading data from a file. Cleaning and organizing text data.

Lesson 4: Play with NumPy arrays—Two of the most important libraries for scientific computing with Python: NumPy and Matplotlib. Importing libraries. NumPy functions to create arrays: linspace, ones, zeros, empty, copy. Array operations. Multidimensional
arrays. Performance advantage of arrays over lists. Drawing 2-D line plots of array data.


Module 2: Take Off With Stats in Python
This learning module builds from a foundation in Python programming to develop data practices and computational problem-solving. Students learn to handle data programmatically, reading data from files, cleaning and organizing data, and performing exploratory data analysis. They use real data, learn to make pretty data visualizations, and gain insight from data.


Lesson 2: Seeing stats in a new light—Continuing with the dataset of canned craft beers, this lesson focuses on visualizing statistics. For quantitative data: histograms and box plots; for categorical data: bar plots. Visualizing multiple data with scatter plots and bubble charts.

Lesson 3: Lead in lipstick—A full worked example using what you learned in lessons 1 and 2: using data from studies by the US Food and Drug Administration on the lead content in lipstick, we fact-check alarming news headlines. Based on Prof. Kristin Sainani’s lecture, “Exploring real data: lead in lipstick,” of her Stanford Online course “Statistics in Medicine.”

Lesson 4: Life expectancy and wealth—A deeper dive into pandas for data analysis, using data of life expectancy and gross-domestic product (income) per capita over time, for various countries across the world. Grouping data for analysis and dataframe manipulation.

Module 3: Tour the Dynamics of Change and Motion With Python
This module builds from a foundation in Python programming to develop modeling and simulation practices, and computational problem-solving. Students learn to capture motion from images and videos, compute velocity and acceleration from position data, obtain velocity and position from accelerometer data, and study differential models of mechanical vibrations.


Lesson 2: Step to the future—Computing velocity and position from accelerometer data: a roller-coaster ride. Using the subplot() function to draw more than one plot in the same figure. Euler’s method for initial-value problems and Taylor expansion showing first-order accuracy. The second-order differential model for an object in free fall written as two first-order differential equations, leading to a vector form. General design of a code to solve ordinary differential equations (ODEs). Application to free fall of a tennis ball and comparison with experimental data. Improved model accounting for air resistance.

Lesson 3: Get with the oscillations—Differential model of a spring-mass system without friction: state vector and system in vector form. Amplitude growth with Euler’s method on oscillatory systems, and the fix: Euler–Cromer method (semi-implicit Euler). Numerically observed order of accuracy using a convergence plot: numerical error with different time increments. Modified Euler’s method and observed order of accuracy.

Lesson 4: Bird’s-eye view of mechanical vibrations—General spring-mass systems with damping and a driving force, revealing a variety of behaviors. Presents a powerful new method to study dynamical systems based on visualizing direction fields and trajectories in the phase plane.
Module 4: Land on Vector Spaces With Python

This module applies Python and core numerical libraries (NumPy, SymPy, Matplotlib) to explore the foundations of linear algebra, with a geometrical and practical approach. Students learn to view matrices as linear transformations of vectors, and develop intuition about their role in linear systems of equations. Playing with transformations, students understand eigenvalues and eigenvectors, and discover matrix decomposition. We use Python to compute all the eigenthings and apply them to population models in ecology, Markov Chains, and the Google Page Rank algorithm. Students learn about singular value decomposition and its application to image compression, least squares problems, and linear regression.

Lesson 1: Transform all the vectors—What is a vector? The physicist’s view versus the computer scientist’s view. Fundamental vector operations: visualizing vector addition and multiplication by a scalar. Intuitive presentation of basis vectors, linear combination and span. What is a matrix? A matrix as a linear transformation mapping a vector in one space to another space. Visualizing linear transformations. Matrix-vector multiplication: a linear combination of the matrix columns. Some special transformations: rotation, shear, scaling. Matrix-matrix multiplication: a composition of two linear transformations. Idea of inverse of a matrix as a transformation that takes vectors back to where they came from.

Lesson 2: The matrix is everywhere—A matrix is a linear transformation: visualize it. Norm of a vector. A matrix maps a circle to an ellipse: visualize it. A vector that does not change direction after a linear transformation is an eigenvector of the matrix. A matrix is a system of equations: visualize it (row perspective). Inconsistent and underdetermined systems. A matrix is a change of basis: visualize it. An inverse of that matrix will change the vector’s coordinates back to the original basis. Matrices in three-dimensional space: linear transformations in 3D; 3D systems of linear equations; dimension and rank. Visualize the transformations of rank-deficient matrices.


Lesson 4: Stick to the essentials: SVD—Geometrical interpretation of singular value decomposition (SVD). While eigendecomposition is a combination of change of basis and stretching, SVD is a combination of rotation and stretching, which can be treated as a generalization of eigendecomposition. Example: SVD in image compression. A 2D image can be represented as an array, where each pixel is an element of the array. By applying SVD and dropping smaller singular values, we can reconstruct the original image at a lower computational and memory cost. Nonsquare matrices: SVD in general; pseudoinverse. Application to linear least squares; linear regression with SVD.

HYBRID TO ONLINE WITH JUPYTER-FIRST COURSE DESIGN

“Jupyter first” alludes to the idea of developing a course first as a set of Jupyter notebooks, then building both an online course and an on-campus learning experience based on those notebooks. For the Engineers Code series, I am creating an online minicourse for each module, in the format of a “massive open online course” (MOOC), using the Open edX course platform. The site is found at http://openedx.seas.gwu.edu, and anyone can register and enroll in the online courses, at no cost.

Open edX is a full-featured open-source platform for online courses, used by millions of learners via the edX consortium, large MOOC platforms abroad (France, China, Spain, and others), and institutional deployments. It allows for third-party extensions with its XBlock specification. With outside technical partners, we developed ways to integrate both content and assessments based on
Jupyter into an online course in Open edX. We have contributed two XBlocks to build courses based on Jupyter: the Viewer and a Jupyter Grader for autograded student assignments (both released as open source). With the Jupyter Viewer XBlock, a course designer can build learning sequences with content pulled dynamically from a public Jupyter notebook (e.g., on a GitHub repository). Jupyter-first courses can be written using an open development model (like any open-source software project), collaboratively and under version control. Once the material is ready, the course builder can create a MOOC-style course in Open edX, pulling the content from the notebooks without duplication in the course platform. (Note that Open edX, itself, does not provide version control of course content.) Moreover, the full richness of presentation in a Jupyter notebook’s rendering is available for display inside the course: formatted equations, syntax-highlighted code, output from computations, data visualizations, etc. (see Figure 1). The code repository for the XBlock is at https://github.com/ibleducation/jupyter-viewer-xblock, and is open source under a BSD-3 license.

Graded Jupyter Notebook XBlock: An instructor creates an assignment using the nbgrader Jupyter extension, then can insert a graded subsection building a Jupyter-based course in Open edX using our XBlocks: Jupyter Notebook Viewer XBlock: From any public Jupyter notebook (e.g., in a public repository on GitHub), pull content into a course learning sequence using the notebook URL (dynamic content). Use optional start and end marks (any string from the first cell to include, and the first cell to exclude) to break a long notebook into unit-sized parts. This allows course authors to develop their course content as Jupyter notebooks, and to build learning sequences reusing that content, without duplication. It also has the added benefit that the development of the material can be hosted on a version-controlled repository. (Open edX, itself, does not provide version control of course content.) Moreover, the full richness of presentation in a Jupyter notebook’s rendering is available for display inside the course: formatted equations, syntax-highlighted code, output from computations, data visualizations, etc. (see Figure 1). The code repository for the XBlock is at https://github.com/ibleducation/jupyter-viewer-xblock, and is open source under a BSD-3 license.

**Summary of the Jupyter XBlocks:** Following is a short description of an instructor’s experience building a Jupyter-based course in Open edX using our XBlocks: Jupyter Notebook Viewer XBlock: From any public Jupyter notebook (e.g., in a public repository on GitHub), pull content into a course learning sequence using the notebook URL (dynamic content). Use optional start and end marks (any string from the first cell to include, and the first cell to exclude) to break a long notebook into unit-sized parts. This allows course authors to develop their course content as Jupyter notebooks, and to build learning sequences reusing that content, without duplication. It also has the added benefit that the development of the material can be hosted on a version-controlled repository. (Open edX, itself, does not provide version control of course content.) Moreover, the full richness of presentation in a Jupyter notebook’s rendering is available for display inside the course: formatted equations, syntax-highlighted code, output from computations, data visualizations, etc. (see Figure 1). The code repository for the XBlock is at https://github.com/ibleducation/jupyter-viewer-xblock, and is open source under a BSD-3 license.
in Open edX that will deliver this assignment (as a download), autograde the student’s uploaded solution, and record the student’s score in the gradebook. The XBlock instantiates a Docker container with all the required dependencies, runs nbgrader on the student-uploaded notebook, and displays immediate feedback to the student in the form of a score table. The code repository for the XBlock is at https://github.com/ibleducation/jupyter-edx-grader-xblock, and is open source under a BSD-3 license.

Note that the course platform displays learning sequences with Jupyter-based content, and can autograde assignments made in Jupyter, but it does not provide interactive computing in cloud resources. Our Open edX platform is open to anyone for registration, and it is not economically feasible for us to offer cloud computing to the public. Students registered in our regular on-campus courses, or attending the Python camps in the Library, have access to a Jupyter Hub instance, for all their interactive computing needs. Learners following our courses from around the world can install Jupyter in their personal computers, or use one of the public cloud Jupyter services, like Binder (free, by Project Jupyter), CoCalc (commercial, with a free tier), or others.

**EVALUATION**

We have some limited evaluation to report, based on student surveys conducted independently of the instructor, which targeted not the effectiveness of the learning modules but rather the impact they may have in student attitudes to open culture. (Our application to the Institutional Review Board, IRB, in the Fall 2017 received an exempt determination.) During the first semester-long course using the (first three) learning modules, we carried out a qualitative study on how a new OER medium, Jupyter Notebooks, may (or may not) impact attitudes of undergraduate engineering students toward sharing and openness. Various research efforts in the last decade have focused on the impact on student outcomes from using OER. We were interested in whether our use of Jupyter, a new genre for OER, may also influence students in their attitudes and capacities for collaboration, community involvement, and open practices. During classroom discussions, we often emphasized the open-source nature of the tools we used, and promoted the idea that students could adopt these tools for any initiatives, during and after their studies. Out of a cohort of 52 students that semester (Fall 2017), 16 voluntarily answered a survey, and 6 agreed to be interviewed by a member of staff in the GW eDesign unit. The survey asked students their opinion about the online nature of the course materials, compared with traditional textbooks: 9 out of 16 students said they preferred the online presentation, three said they liked it less (three had no preference, one did not answer). The presurvey also aimed to sample a baseline of attitudes toward sharing and openness, via an open-ended question. Students’ answers reflected equally pro-sharing and anti-sharing attitudes, the latter mostly referring to others taking “credit” for their work. Certain cultural barriers persist that inhibit openness, it seems: some students’ expectation of having a textbook, and a culture of competition and fixation on grades. We concluded that a single semester-long course is not enough time to make a change in culture. Open educational practices should be threaded through several courses, and start early. Collaboration with other instructors is needed to reinforce ideas, skills, attitudes across more than one course.

In the Spring 2019 semester, the introductory modules were taught as a new first-year course (as part of a two-course series). Quantitative evaluation with this cohort targeted students’ perceived gains in computing skills, and changes in attitudes toward coding. Out of 48 students taking the course, 23 responded to a survey (69% male and 31% female). At the end of the course, the responses to “How prepared are you for learning to code?” increased from an average of 5.4 to 7.4 (on a 0–10 scale). The survey also asked several questions on perceptions about using coding skills after graduation, most of which displayed a modest increase. On the prompt “I plan to use coding in my career after graduation,” the average of responses increased from 6.2 to 7.3.

We also carried out a short survey on perceived usefulness, targeting the first cohort of students that used the learning modules in Fall 2017, when they were in their second year. They are Seniors now, and have the benefit of time in
assessing the usefulness of the materials. The survey just had three questions on a scale from 1 to 5 (from “not at all” to “a lot”), as follows.

1) Do you find yourself using these computational tools in other courses later?
2) Have you learned more Python techniques over time?
3) Do you think you will use Python/Jupyter/computing-in-general in your future career?

Thirty-two out of 50 possible students responded this survey (two students had left the university). To the first question (used the computational tools in other courses later), 78% chose 5 (“a lot”) and an additional 19% chose 4. Only one student chose 1 (“not at all”). To the second question (learned more over time), 37% chose 5, and 41% chose 4.

The evaluation is modest, but our focus has not been to conduct education research, but rather to create the learning materials according to our design specifications, and to develop complementary technology (Jupyter viewer and auto-grader XBlocks) to provide a blended learning experience.

CONCLUSION

This article for the CiSE Special Issue on Computational Science and Engineering Education presents our collection of learning modules for undergraduate engineering students, aimed at developing foundational computing and data skills. These modules are developed in the open-source model (on GitHub), designed to be reused, and shared under standard public licenses allowing redistribution and revision by users to fit their needs. Our long-term vision is to collaboratively develop several more modules, aiming to populate the engineering curriculum with computational content. Ultimately, the education of STEM graduates can be indelibly transformed by computing and data skills becoming *infrastructural*, when all learners are led and supported to become proficient in them, and to apply them in authentic technical contexts. This is the defining feature of a *literacy*: a socially widespread deployment of skills and capabilities that become material support to achieve valued intellectual ends. Knowing how to read and write, that is, the conventional meaning of literacy, is not only highly valued in society, but the bedrock of all education. Similarly, computing and data skills are today indispensable to almost all STEM fields, and form the basis of a new literacy.

The *Engineers Code* project implements key concepts and design principles, distilled from several years developing instructional materials using Python and Jupyter, and informed by the education literature. It is not education research, but rather an implementation project. The education-research literature supports our design principles, however. Active learning is superior to lectures in both comprehension and recollection and we apply it using Jupyter notebooks combined with live coding and other tactics like pair programming. The worked-example effect, the most-studied cognitive-load effect, explains why our design helps students manage the complexity of learning applied computing. Modularization, chunking, interleaving: these are known effective techniques in learning, which we have put to work in this project. Ongoing evaluation is helping us confirm the positive effect on students, and build confidence to share with others interested in adopting our approach.

The project is also producing online “mini-courses” using a MOOC platform (Open edX). In the process, I have refined an approach that I call *Jupyter-first* course development, where a course is first written as a set of Jupyter notebooks, and the learning sequence in the online course platform pulls from this content dynamically. To achieve it, we created with technical partners the third-party extension (XBlock) to embed content from a public Jupyter notebook in a course. We also created an extension to allow autograding of student assignments written as Jupyter notebooks. All this technology has been released to the public and is open source. The online modules are also open for anyone to enroll and follow at their own pace, and could even be assigned by other instructors to complement their courses. We know of one instructor at another university who did so, and also have collaborated with the GW Libraries to help them offer Python camps using a combination of our first module and parts of the second. They condensed the face-to-face learning to three or four days, and learners complete the autograded assignment in the online platform (Open edX) after the camp to receive a certificate of completion.
These Python camps are in high demand, and the GW Libraries reported that registrations fill all available seats in under eight hours from announcing the event! Three camps were held in 2019, and the Library plans to continue offering them in 2020. We have also started to support these informal learners, and the learners in the official GW courses, via Study Hall sessions in the Library, staffed by undergraduate student tutors. Our vision is to help coalesce an active learning community in the university for learning foundations of programming in applied contexts.

ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Cyberinfrastructure (OAC), National Science Foundation under Grant #1730170. The student surveys and interviews during Fall 2017 were conducted by Tara Lifland, then an instructional designer at the George Washington University eDesign Shop and OER Research Fellow (http://openedgroup.org/fellowship) of the Open Education Group. The surveys during the Spring 2019 were conducted by Prof. Ryan Watkins from the GW School of Graduate Education and Human Development. The GW Libraries Python Camps are led by Megan Potterbusch and Laura Wrubel. Several graduate students in Dr. Barba’s group have contributed to her educational initiatives, and the development of learning modules (for this project and its predecessors): Natalia C. Clementi, Pi-Yueh Chuang, Gilbert Forsyth, Olivier Mesnard, and Tingyu Wang. With special and heartfelt gratitude toward all the contributors to Project Jupyter and its ecosystem of tools for education. Project Jupyter is a fiscally sponsored project of NumFOCUS (https://numfocus.org), a 501(c)(3) U.S. non-profit with a mission to promote open practices in research, data, and scientific computing.

REFERENCES


Lorena A. Barba is a Professor of mechanical and aerospace engineering with the George Washington University. Her research interests include computational fluid dynamics, biophysics, and high-performance computing. She is Editor-in-Chief of Computing in Science & Engineering (CiSE), Co-Editor of the CiSE Reproducible Research Track, Associate Editor for The ReScience Journal, Associate Editor-in-Chief of the Journal of Open Source Software, and Editor-in-Chief of the Journal of Open Source Education. She received the Ph.D. degree in aeronautics from the California Institute of Technology, Pasadena, CA, USA. For more information, please see https://lorenabarba.com. Contact her at labarba@gwu.edu.
Research-Based Education on a Master of Science Degree in Professional Software Development

Cathryn Peoples
Ulster University

Abstract—This article contributes to the narrowly-investigated field of research-based assessment. Research-based assessment supports student learning by offering choice in how it takes place. It is not widely offered, however, for reasons which can include the challenge of marking outputs consistently, and the importance of ensuring that students engage early with the task. The approach presented in this article exploits this technique, and additionally merges it with authentic assessment, where students are involved in the assessment design. The study confirms effectiveness of the approach through the mark profile, in spite of all students not engaging with it at the earliest opportunity. The study also identifies how students became more competent with research-based assessment when reflecting on feedback for a similar piece previously assessed.

The research-based education describes an application of the results of research to facilitate and supplement student learning. When research-based education is used, students are required to use material beyond that delivered during a lecture or made available through the virtual learning environment (VLE) by their tutor. Instead, they are expected to be proactive in both identifying and assimilating the information required.
The importance of research-based learning is reinforced in the literature, such as in Healy and Jenkins.\textsuperscript{1} Also referred to as inquiry-based approaches, it is considered to be important to push students to the limit of their knowledge, and to act in the role of a knowledge producer, in addition to being a knowledge consumer. It requires learners to actively identify material that will supplement their knowledge. With active learning, students are not just only listening but are also engaged in solving problems.

Research-based education can benefit students in several ways. As a result of research-based education, students may:

- gain experience in selecting learning material;
- become familiar with the concept of plagiarism and how to avoid it;
- become familiar with techniques to reference externally-sourced material and to use in-line citation;
- become more critical in the subject area through being offered a choice in the teaching and learning material;
- become more effective in time and project management through having to be responsible for sourcing supporting material and gaining new knowledge in advance of beginning their task.

Teaching through research also benefits staff in the following.

- Engagement of students throughout the teaching period may be more likely, when they are required to actively source learning materials before beginning a task.
- Students may be more likely to work in a unique manner, the style of which could be applied by teaching staff in later teaching experiences.
- Project specifications given by teaching staff may be vaguer by presenting general topics, such as "smart city" or "Internet of Things," and allowing students to identify the specific aspect to investigate within the domain.
- Greater diversity in submissions may reduce the potential of monotony when marking in large classes.

Research-based learning has been applied by other researchers to support students of software engineering. Applied by da Cunha\textsuperscript{2} under the name of project-based learning, students were required to gather user requirements, and then to implement two prototypes of the system. Suitability of the systems implemented was dependent on the effectiveness of the research conducted as part of the requirements gathering.

In a contrasting approach, research-based learning has been applied to a cohort of students on a Master of Science (M.Sc.) degree in Professional Software Development at Ulster University. In addition to the benefits noted above for both students and staff, it has been widely recognized that there are limited opportunities for students to develop writing skills in STEM subjects.\textsuperscript{1} It was therefore based on the combination of these reasons that research-based assessments were used as part of the teaching and learning process for this student cohort. In this article, experiences with the research-based assessment on the M.Sc. degree are presented. The effectiveness of student engagement with the process is considered, in terms of both the quality of the assessment submissions and the marks awarded.

LITERATURE REVIEW

Research-Based Learning

Research-based learning is a form of flexible learning, as described by the Higher Education Academy to “… [empower] students by offering them choices in how, what, when and where they learn …”.\textsuperscript{3} While given a broad topic on which they will be assessed, together with their learning objectives, responsibility is delegated to students to search for and select material to complete the task. A study by Christe \textit{et al.},\textsuperscript{4} as an example of research-based assessment, reports on an independent task given to the undergraduate students from under-represented groups that include women and members of minority ethnic groups. The students were required to review at least three scientific articles, and to use the information to define a research proposal. This task was viewed as giving students, "open-ended problem solving experiences," with a focus on the student-librarian partnerships in their development of research ideas.

Research-based learning is also explored in Shaban and Abdulwahed,\textsuperscript{5} with the objective of
the task reported here being to submit research papers to international conferences and journals for peer review. The problems explored involve real-life research scenarios, described as covering a range of different engineering disciplines. Students worked in groups of a maximum of three, with each team member having different responsibilities that were rotated during the research period. At the end of each phase of the research (e.g., defining the problem, solving the problem, interpreting results, and compiling findings), outputs were documented in a research portfolio. In addition to a publishable output, students were also assessed through a presentation.

Wu et al. 6 used an engineered platform to support students with a research-based assignment. A mobile newspaper system is used as a repository of research resources. Students select research topics in line with their personal interests. The platform provides research materials to the student in response, which can subsequently be analyzed and used to complete their assignment, or be used as a springboard for students to continue their search for further supporting material. Students are required to interact with the system after the materials are initially provided by recording their progress; the system analyzes the requests that students send out, and subsequently recommends further study resources based on the students’ characteristics.

There are a limited number of academic papers in the field of engineering, which report the use of research-based assessment to support student learning. The research presented in the papers above is in line with the concept described in this article, and provide comparative approaches in which the technique is used. This article seeks to contribute to this narrowly investigated field.

Research-Based STEM Assessment in Relation to the Gardner Multiple Intelligences Learning Theory
These include:

Linguistic: The ability to use spoken or written words.
Logical-Mathematical: Inductive and deductive thinking and reasoning abilities, logic, as well as the use of numbers and abstract pattern recognition.

Visual-Spatial: The ability to mentally visualize objects and spatial dimensions.
Body-Kinesthetic: The wisdom of the body and the ability to control physical motion.
Musical-Rhythmic: The ability to master music as well as rhythms, tones and beats.
Interpersonal: The ability to communicate effectively with other people and to be able to develop relationships.
Intrapersonal: The ability to understand one’s own emotions, motivations, inner states of being, and self-reflection.

Where a student is considered to not have fully engaged with the assessment, which is an important consideration in the case of the research-based assessment, this may be because it has not targeted the specific way(s) which they reach an understanding. For an assessment to resonate with as many students as possible, it is important to accommodate as many of the intelligences as is reasonable. In the assessment of software engineering knowledge, there are a limited number of Gardner’s intelligences that may reasonably be incorporated in any task; Musical-rhythmic, Body-kinesthetic, and Visual-spatial, as examples, are unlikely to be viable options.

It can be more plausible, however, to appreciate how a research-based STEM learning task can be constructed such that it accommodates: Linguistic, Logical-Mathematical, Interpersonal, and Intrapersonal intelligences. Gardner identifies that our culture focuses on linguistic and logical-mathematical intelligence. The basis of this theory is that attention should also be given to learners who are not as competent in a linguistic or logical-mathematical way, and therefore the feature mechanisms that reach out to those who have strengths in the other values, such as inter- or intrapersonal ability for example. The research-based approach to assessment meets the needs of students with linguistic and logical-mathematical intelligences. It also corresponds with the capabilities of learners who are self-smart, with intrapersonal awareness, through being able to harness personal qualities, such as motivation to work independently and time management ability. Good interpersonal skills can also be exploited during research-based assessments, with collaborative learning across the group being one technique to support student performance. It was
therefore with this understanding that the research-based assignment was set.

RESEARCH METHODOLOGY

Class Profile

The research-based assignments that are the focus of this research were given to a cohort of students on a Master of Science (M.Sc.) degree in Professional Software Development at Ulster University. This is a conversion degree into Information Technology (IT) for students from non-IT backgrounds. Modules taken as part of the degree include: computer hardware, operating systems, software development, data structures, databases, concurrent systems, and mobile devices and applications. For entry onto the degree, it is essential that students do not have an academic background in Computing. Students therefore have a diverse academic profile, including Transport Management, Russian Language and Literature, Theology, and Criminology. As pre-entry requirements onto the degree, students must have a minimum of a 2:2 degree. The breakdown of pre-entry qualifications across the cohort studied is: 11 first class students, 31 second (higher) class students, and 23 second (lower) class students. There were originally 65 students on the degree program, although several left the course throughout the year.

Research-Based Approach to Teaching and Learning

A research-based approach to teaching and learning was used by the author for several of the modules studied on the M.Sc. degree across two years. In teaching through research, students are examined on more than their knowledge of the subjects—additionally, their understanding, original thinking, and ability to articulate a technical argument. Applied across a range of subjects, as summarized in Table 1—data structures, computer hardware, and operating systems—the research-based assignment presented in this article is specific to the operating systems module.

Operating Systems Module on the M.Sc. Programme

As part of the assessment suite for the operating systems module, students were asked to prepare a report using the title, “Ability of the Linux Operating System to Support Online Game Deployment.” The title of the research task was selected to accommodate several goals:

- exposure to state-of-the-art technology for students from non-IT backgrounds;
- a topic which could be assumed to be of general interest to students (online gaming) and which would ideally therefore foster engagement across the class;
- a timely research question which may help students to form ideas about their final dissertation projects.

Based on their research, students were required to discuss:

- operational differences between the Linux operating system and another operating system of their choice;
- operational and performance QoS requirements of online games;
- characteristics and performance abilities which enable the Linux operating system to support the requirements of online games;
- the changing operational environments in which online games are run, which take into account cloud-based technology and virtualization;
- ability of Linux to support online games in the future.

In the completion of this assignment, students were required to research aspects that include the Linux OS, the Windows or Mac operating systems, QoS requirements for game play,

---

<table>
<thead>
<tr>
<th>Module</th>
<th>Assignment title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data structures</td>
<td>Design of Data Structures and Algorithms to Support a Social Networking Website</td>
</tr>
<tr>
<td>Computer hardware</td>
<td>Compare and Contrast the Hardware Deployed in the Univac and the Google Data Centres</td>
</tr>
<tr>
<td>Operating systems</td>
<td>Benchmarking the Resource Footprint of a Linux Ubuntu Server 12.04 LTS Operating System deployed in a VMware Virtual Machine</td>
</tr>
<tr>
<td>Operating systems</td>
<td>Ability of the Linux Operating System to Support Online Game Deployment</td>
</tr>
</tbody>
</table>
characteristics of alternative operational environments in which games may be played and assessment of how they influence game play, and future expectations with regard to playing games online. Based on their research, students were also required to reach a conclusion with regard to how suitably they felt that Linux can support the requirements of online games.

The assignment was worth 30% of the overall module marks (of a 100% coursework module). A pass mark of 50% was required for this module. Marks for this piece of coursework were allocated for:

- uploading a draft of the report in advance of the submission deadline (5%);
- scene-setting and defining the problem domain (10%);
- selecting and using related work from the field (10%);
- discussing the suitability of the Linux operating system for supporting online games, both now and in the future (50%);
- overall presentation (10%);
- originality (10%);
- additionally, students were required to indicate one further criterion on which they wished their work to be assessed (5%), to be described in the Appendix in under 50 words.

Awarding a proportion of the overall marks for uploading a draft in advance was adopted to understand the extent to which students engaged with the task at an early stage. The selection and use of externally-sourced information helped to indicate how well students were able to carry out effective research, and applying it, through their discussion.

The submission was required to be presented on a single double-column page to give students the experience of preparing a technical “paper” in a format suitable for publication. On the second page of the report, students could include figures, any supplementary Appendices, references to the related work discussed in the report, and specification of the criterion on which they wished to be assessed. Additionally, they were also asked to indicate the way in which they applied the feedback received for an earlier similar piece of work, which involved a writing task based on research into a state-of-the-art technology. In preparation for this task and in an attempt to help them to engage, students were provided with four references\textsuperscript{8–11} which they were required to, at a minimum, include in their report, and use to define the background to the problem.

This piece of coursework is applicable to the module learning objectives as it encourages students to consider the role that the operating system plays in supporting the requirements of a specific application type. The essential aims of the operating systems module on the M.Sc. program are to:

1. introduce the role and function of modern operating systems;
2. provide students with the ability to compare and analyze modern alternatives for resource management in computer systems;
3. provide students with the ability to develop shell scripts;
4. provide students with an understanding of the use of built-in utilities for OS programming.

This assignment allows students to analyze the ability of different operating systems to support the different resource management requirements in systems through understanding the different abilities of each to support the chosen application requirements. The benefits of teaching through research for the operating systems module include that students can become aware of the fact that technology must be enabled to support the requirements of applications, and that different operating systems have differing ability to do so. Furthermore, by offering students a choice in the technologies that they can refer to in the completion of their assignment, they can become critical in the capabilities, or limitations, of each.

Within the suite of three assessments for the module, the one presented in this article was the first piece, distributed in Week 2 of the six-week module period, to be returned by the end of Week 3; students therefore had nine days to complete the work. This assessment was selected as an early piece of work to introduce students to the concept of operating systems.
RESEARCH FINDINGS

Student Participation in the Research-Based Assignment: Identifying an Additional Assessment Criterion

To encourage students to become active participants in their learning, as an essential component of research-based learning, they were given the opportunity to indicate one criterion on which they wished their work to be assessed; this was worth up to a maximum of 5% of the overall assignment mark. Clearly, this technique is not specific to research-based assessment, but was applied to give students a feeling of control over their learning, and to support them in prioritizing those aspects which are most important to what they want to learn from the module. Application of such a metric is possible in a research-based assignment, given the relatively open, and potentially, varied, nature of responses that can be produced—a “right” or “wrong” answer does not exist per se, and there is a scope to be flexible in how and where marks are awarded.

A selection of the responses received from students are presented in Table 2.

Responses that were rewarded more highly than others demonstrate that the students had processed the assignment specification, and had given careful consideration to this requirement—the categories identified were not a repeat of requirements already defined in the assignment specification. A reasonable criterion includes, for example, demonstrating understanding of how low level processes of the OS affect high level user interaction with the application (#3). Another category rewarded was the tone that they had incorporated in the writing style (#6). These aspects were not obvious requirements from the specification, and demonstrated that the students had carefully considered how they might contribute something further in their submission.

Other students, however, were less successful in identifying an additional area in which they wished to be assessed. While one student wished to be assessed on her inclusion of source material (#4), which would have been a viable category for reward, she did not provide any additional material beyond that available in the VLE; minimal marks only could therefore be awarded in this case. Other criterion given indicated that students had not carefully read the marking scheme. For example, one student wished to be rewarded for, “understanding the requirements of an operating system for online gaming.” (#9) This criterion could not be fully assessed as an additional one, as it was on this aspect that students would be assessed overall for their submission. Similarly,

---

Table 2. Assessment criterion defined by student.

<table>
<thead>
<tr>
<th>ID</th>
<th>Criterion</th>
<th>Mark awarded (total 5%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Want to be awarded marks on the application of feedback from Assignment One.</td>
<td>3%</td>
</tr>
<tr>
<td>2</td>
<td>Ability to understand, assimilate, and discuss academic studies such as the one on performance.</td>
<td>3%</td>
</tr>
<tr>
<td>3</td>
<td>I wish to have the subjective criteria considered on the balance of understanding how low-level process affects high level user interaction.</td>
<td>3%</td>
</tr>
<tr>
<td>4</td>
<td>Additional marks could be given for readability and quality of English. Also, inclusion of the source material.</td>
<td>2.5%</td>
</tr>
<tr>
<td>5</td>
<td>I would like my work to be assessed on the wide range of issues covered and how well documented I addressed the capabilities and functions of the Linux operating system as a whole.</td>
<td>2.5%</td>
</tr>
<tr>
<td>6</td>
<td>I would like this assignment to be marked in terms of its voice. I have attempted to add some personality and a particular tone throughout the piece.</td>
<td>3.5%</td>
</tr>
<tr>
<td>7</td>
<td>I wish for my work to be assessed on my knowledge of online gaming and the technology that is available at the moment for gaming devices.</td>
<td>2.5%</td>
</tr>
<tr>
<td>8</td>
<td>Selection of recent additional references.</td>
<td>3%</td>
</tr>
<tr>
<td>9</td>
<td>Understanding of the requirements of an operating system for online gaming.</td>
<td>1%</td>
</tr>
<tr>
<td>10</td>
<td>I wish to be assessed on my ability to answer the posed question throughout the report, with regards to knowledge researched and taught in class.</td>
<td>1%</td>
</tr>
</tbody>
</table>
response #10 also could not be fully assessed, given the fact that it was this criterion, their response to the question asked, on which the overall piece of work was being marked.

The result of asking students how they would like to be marked is an important finding: Where students are under-performing, it is possible that they are doing so because they do not fully understand from the specification how marks are being allocated, and therefore fail to tackle them appropriately in their submission. It may be appropriate to complete such an activity as an early hand-in, with preliminary feedback given to redirect students along the correct path, when needed.

Table 3. Use of prior feedback.

<table>
<thead>
<tr>
<th>ID</th>
<th>Use of prior feedback</th>
<th>Mark awarded</th>
<th>Mark awarded for previous assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A word count has been included. More academic journals and reliable recent sources of information have been used. Consideration has been given to the use of better grammar, structure and writing style.</td>
<td>67%</td>
<td>60%</td>
</tr>
<tr>
<td>2</td>
<td>I must write as if the reader does not know anything about the work that has been written, so when writing this new piece of work, I tried to be clear and precise in my writing I have tried to incorporate appropriate quotes, articles or evidence to support the statements I have made.</td>
<td>62%</td>
<td>53%</td>
</tr>
<tr>
<td>3</td>
<td>It was noted previously that the ‘story’ did not flow well throughout the assignment. For this piece of coursework, care was taken to relate each section into the next, to provide continuity between sectionalized parts.</td>
<td>67%</td>
<td>56%</td>
</tr>
<tr>
<td>4</td>
<td>I have tried to provide a better explanation in my conclusion.</td>
<td>54%</td>
<td>48%</td>
</tr>
<tr>
<td>5</td>
<td>I outlined the objective of the paper in the introduction, describing what will be found in the report.</td>
<td>68%</td>
<td>74%</td>
</tr>
<tr>
<td>6</td>
<td>I have tried to be more diligent in my efforts at editing and scrutinizing my work. I have also seen from previous works that referencing has been somewhat lacking. I have tried in this essay to backup as many statements as possible.</td>
<td>66%</td>
<td>61%</td>
</tr>
<tr>
<td>7</td>
<td>Removed American English. Provided references where appropriate. Removed unnecessary paragraphs. Used up-to-date material for references. Tried to provide a clearer background to the report in the introduction.</td>
<td>77%</td>
<td>71%</td>
</tr>
<tr>
<td>8</td>
<td>I tried to include relevant specific detail as much as I could within the limited space. All of my own references were from 2014, to address the use of old references. A word count is included. American English is avoided.</td>
<td>65%</td>
<td>65%</td>
</tr>
<tr>
<td>9</td>
<td>I also needed to do more reading around the area; I have been improving on this by reading the supplied material and other online articles to get a better understanding of the topic area.</td>
<td>57%</td>
<td>56%</td>
</tr>
</tbody>
</table>

Student Participation in the Research-Based Assignment: Identifying Application of Feedback Previously Given

Students were also required to indicate the way(s) that they had used feedback received for a similar research-based assignment (see Table 3). A mark was not allocated for providing this information, but was used in the evaluation of the effectiveness of the teaching and learning process through feedback given for this cohort of students.

By asking students to articulate how they had applied the feedback, it was hoped that attention had been given to those aspects that were lacking previously. The mark profile between two similar pieces of work, in the majority of cases, indicates that this is the case. For example, in response #9, the student acknowledged that their previous feedback advised that they read more around the subject area. It is interesting to note that this student did not, however, include any references additional to those provided in the assignment specification, and their mark only increased slightly from the earlier piece. In other more positive examples, such as #2, the student has tried to write with more clarity and include references to supporting sources of literature. The student writing response #1 has also included more sources of information to support the arguments that they are making.

Using research-based assessment is particularly feasible for classes of mature students,
such as a master's degree program, where students have been through at least one academic experience previously. A number, although not all, have completed dissertations previously; the majority are therefore familiar with the process of finding literature, of citing related work, and of preparing reference lists. Ensuring that their skillset is being expanded through the M.Sc. degree, they are tested in the research-based assessments through the identification of reputable sources relevant to the STEM field, being able to understand the message being delivered, and using it to construct a technical argument.

Student Results When Assessing Using Research-Based Learning

The average result for the piece of coursework was 65%, with a maximum of 77% and minimum of 49% (see Figure 1). Initial teacher-led tasks were carried out to encourage students to engage early with the assignment, helping them to appreciate the scale of the task, to subsequently manage their time more effectively, and to avoid underperforming through not having sufficient time to complete the research necessary. Marks (of up to 5% for a draft submission) were allocated for evidence of early engagement to act as encouragement. Providing that a draft was uploaded and there was evidence of some research work having been completed, the mark was allocated—students therefore secured a mark of 0 or 5 for this aspect of the assignment. Surprisingly, however, only 47 students out of the class of 63 took advantage of the opportunity.

This result demonstrates that a number of students did not engage with the task when it was first disseminated, despite marks being available for doing so. There may be a number of reasons for this: first, it is possible that students did not read the mark scheme closely after the assignment was first set, and may therefore not have been aware of this contribution to the overall grading. Furthermore, students may not have been present in class on the day when the assignment was discussed, and therefore also missed the oral delivery of the information. This result may also highlight some ineffectiveness of the teaching staff in engaging students with the task.

The mark allocated for students identifying the additional criterion on which they wished to be assessed was the most disappointing mark for the assignment, and this element was failed when considered across the class as a whole (see Figure 2), with an average of 2.32% out of the maximum of 5%. Several students did not receive a mark for this category: Some students did not provide an additional criterion; other students provided an unsuitable one. One student, for example, indicated that they wished to be assessed for their ability to demonstrate understanding of the requirements of an operating system for online gaming. This received a mark of 1 out of 5—the student identified the need to select an additional category for marking, but unfortunately, the one selected was already one being assessed. Another student asked to be assessed on her ability to answer the posed question throughout the report. Again, this achieved a low score due to the fact that it is an aspect already being assessed in the general marking scheme.
CONCLUSION

Specific to the M.Sc. degree in Professional Software Development, research-based learning in the form of a writing task is particularly relevant due to the fact that a number of the students had prior academic experiences through which they developed writing skills, with a number coming from an arts or business background. In the midst of the software development tasks, which they were being assigned in parallel, it was believed that it may help to encourage learning in a format that is perhaps more comfortable due to their familiarity of working in this way.

This does not ignore the fact, however, that there are challenges associated with this approach. One particular challenge comes from the perspective of marking: As it can be anticipated that there will be greater diversity between each piece of work submitted, the time to mark each contribution will increase in parallel. This is largely a reflection of the fact of it being a research-based assessment and is difficult to eliminate. Elements of the specific approach presented in this article, such as the criterion defined by each student and discussion of how past feedback has been applied could be removed from the assignment to improve time efficiency when marking. Challenges of research-based teaching also include ensuring that a minimal set of information is presented to support students who do not fully engage with the practice, in the sense of sourcing a thorough set of literature, and ensuring that they are adequately equipped to complete the assignment successfully. While they may be penalized for not participating in the task more thoroughly, they should be able to achieve a basic level of performance. There is also a risk in research-based education that students can become more lost in their learning experience than in nonresearch-based situations. Techniques for encouraging early engagement are therefore particularly important, such as the allocation of marks in the category for an early draft submission and feedback provided on an early check that they have read the coursework specification closely. Research-based education will be more difficult to execute successfully when students delay the task until the last opportunity.

REFERENCES


Cathryn Peoples is currently a part-time research associate with Ulster University, Coleraine, Northern Ireland, and a part-time associate lecturer with the School of Computing and Communication, Open University, Milton Keynes, U.K. Her research interests include cloud management, smart cities, and green IT. She received the Ph.D. degree in delay-tolerant networking from Ulster University in 2009, and is a member of the British Computer Society, IET, ACM, and IEEE. Contact her at c.peoples@ulster.ac.uk.
Visualizations are essential tools that provide insight into the geometrical arrangement of a studied sample, facilitating understanding of chemical and physical properties. The ways to visualize a crystal structure using AViz software were reported previously. Different visualization levels were demonstrated, such as visualization techniques for modeling carbon allotropes and visualization of electron density. For the macromolecular structures in biochemistry, various levels of visualization are critical. Many proteins that have variations in amino acid sequence (primary structure) may still perform similar biological functions. This similarity is attributed to the...
homology in tertiary and quaternary structures (3-D folding). This level of organization of proteins is critical for the function; for this reason, it is highly conserved. Hence, some details about the exact sequence of the amino acids may be omitted in the macrorepresentation of a protein. However, the secondary structure of a protein and the atomistic representation of a ligand-binding site are of a great importance. Since no biochemistry research is held without knowing or investigating the exact structure of an enzyme or a protein, the importance of visualization of structures is one of a primary task.

There is a variety of software for molecular visualization.

**MOTIVATION AND DISCUSSION**

Biochemistry A—CHEM 370 is a first semester course of a two-semester sequence for biochemistry and chemistry majors and is focusing on structural biochemistry and some metabolic processes, such as glycolysis, gluconeogenesis, citric acid cycle, and oxidative phosphorylation. This class is conducted in a traditional setup of three 50-min lectures (50–80 students in a classroom) and one smaller discussion session (25–40 students per session) per week. The focus of the lectures/discussions is to teach biochemistry fundamentals and applications of basic chemistry principles in microdimensions of a biochemistry world and to develop scientific and critical thinking.

One of the topics discussed in this class is a homology alignment. Students were given a truncated sequence of a human hemoglobin alpha chain. The goal of this activity was to understand alignment scoring and use of different statistical algorithms in searching for homology of this sequence with any available sequences in the Basic Local Alignment Search Tool (BLAST) databank. In this activity, students copied the given sequence in the “Enter Query Sequence” field (see Figure 1).

This activity aimed to demonstrate that regardless of some differences in the primary structure of the protein, the tertiary and quaternary structures of the proteins are more conserved. In the next step, the students were sent to the Protein Data Bank to find a homologous protein based on the data collected from the BLAST activity. Students were asked to use the 3-D view applet available on this webpage to visualize the 3-D structure of the protein as well as protein–ligand interaction.

The tasks mentioned above were first steps in making students familiar with the computer visualization of the proteins and ligands, labeling atoms, and amino acid residues. As the semester progressed, students were exposed to a variety of proteins with different structures and biological functions. The origin, chemical structures, and biological functions of ligands were discussed in great details and thoroughly classified as the class moved through the material. The students were divided into project groups of two and were
TECHNICAL DETAILS OF PEDAGOGICAL APPROACH

For the visualization project, students were asked to download and install a molecular graphics program VMD\(^9\) on their computer. It is important to mention that the computer, software, and hardware literacy of junior and senior students of life science disciplines is highly limited. The pedagogy behind this project developed by us was to expose students to a new field and remove a fear of technology. The following instructions were given in order to educate them about hardware of their machines. It is important to present a fragment of the handout given to students to demonstrate the level of the computer education used in this class.

1. Go to the VMD download site at the link
   
   https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD.

2. Select the version of VMD that is appropriate for your hardware. If you have a Mac, there is only one. If you have a PC, it is the Windows Open GL version. If you have an Nvidia CUDA graphics card, try the Windows Open CUDA version. After you double click on the version, follow the instructions to register with the site. Usually the registration of a student goes as undergraduate using the program for class.

3. To start using the program, you can get an installation guide, a user guide, and a tutorial (along with the files needed to work through it) at
   
   https://www.ks.uiuc.edu/Research/vmd/current/docs.html.

4. Pick a protein from the pdb (http://www.rcsb.org/pdb/home/home.do). If you do not have a favorite protein, pick one mentioned in the textbook. It is highly recommended to use an enzyme or complex from any metabolic pathway that will be discussed in this class. Be sure that it has a nonprotein part. This can be a metal ion, a substrate, an inhibitor, or a coenzyme.

5. Make three diagrams of your protein using VMD. One should be the protein shown in New Cartoon so that you can see the secondary structure. Try coloring this one according to the secondary structure. Remember to go to “Colors” on the “Graphics” menu to make the background white. Use “Tachyon” on the “Render” dropdown that is found on the “File” menu to make your picture. Remember to change “vmdscene” on the filename to something that makes sense to you. The bitmap file (’.bmp) will be stored in the VMD program directory, unless you have changed directory (cd command). If you are having trouble finding your bmp file, try using the pwd command to determine the present working directory.

6. The second picture should be one that shows the active site of the molecule. If there is a ligand bound to the molecule, try using the following selection criteria to get a good picture.
   
   a. Double click on the line in the graphics menu for your whole protein. This will cause the current diagram to disappear.
   b. Create a new representation. Select “rename LIG” and show it in “Licorice.” Note that “LIG” can be any nonprotein rename. Click on “Selections” tab on the “Graphical Representations” menu and then on “rename” under the “Keyword” list to get a list of resnames in the molecule.
   c. To get the environment of your “LIG,” create a new representation and then type the following in the “Selected Atoms” space: protein and same resid as within 4 of resname LIG. This should give you a diagram of the residues that have at least one atom within 4 A of you LIG molecule.

7. For the third picture, just try to make something that you think is pretty using the different drawing methods and coloring methods under the “Draw style” tab on the “Graphical
Representations” menu. This will be graded on how good it looks and its originality.

8. For all three pictures, be sure that they are large enough to see the structure that you are illustrating. Make sure they are about 6 in wide. The pictures should be inserted into your ppt doc. Actually render the pictures. Do not use screenshots. Some examples are attached at the end of this write-up.

9. Get basic information about your report from the pdb file, which can be read using PowerPoint or Notepad since it is a text file. In the 10–15 min PowerPoint presentation, you want to report a brief tutorial (you want to use vmd software and PowerPoint) how you got your third image of the presented images, the four character pdbid, the bibliographic reference to where the structure was reported, the name of the protein, its source (cow, pig, etc.), the resolution of the structure, and a brief statement about its function (for this reason, you have to be at least slightly familiar with the metabolic pathways). This should include the reaction catalyzed if it is an enzyme. You will have to look up this information either in the biochemistry textbook or on the web or in the introduction to the article on the structure of the protein. Put all of this information in the PowerPoint presentation. There are example pictures in Figure 2. Upload your PowerPoint presentation on Sakai >Assignments. Make the name of the file your last name + VMD.ppt.

**CONCLUSION**

Historically, there is a significant gap between the level of skills and knowledge in major discipline and technological literacy of students in life science disciplines. This gap is narrowing as students are exposed to research. One of the goals of the current project was to indirectly expose students to differences in hardware and software and their use in life science disciplines. For this reason, it was not sufficient to just visualize a protein. The handout partially presented in this article was modified from the one previously developed by Prof. Keneth Olsen teaching similar class in our department. Dr. Olsen used a similar activity as one of the 50-min exercises provided to students during one 50-min discussion session or as a homework assignment. Our approach is using similar scientific tool but completely different in the pedagogical methodology.

The pedagogical goal of the project was an integration of the following:

1) development of a computational literacy;
2) visualization of biological macromolecules using available biochemistry computational databases;
3) using the aforementioned skills in getting more information about the protein of interest;

![Figure 2. Upper image: Tertiary structure of myoglobin using New Cartoon representation, demonstrating the secondary structure. Bottom image: Hem prosthetic group using Licorice representation. These images were made with VMD/NAMD/BioCoRE/JMV/other software support. VMD/NAMD/BioCoRE/JMV/other is developed with NIH support by the Theoretical and Computational Biophysics group at the Beckman Institute, University of Illinois at Urbana-Champaign. The crystal structure is reported by Smerdon et al.](image-url)
4) developing collaboration and professional communication skills;
5) presenting the results in a professional way; and
6) following the formats and deadlines.

REFERENCES


Polina Pine is an Advanced Lecturer with the Department of Chemistry and Biochemistry, Loyola University Chicago, Chicago, IL, USA. She teaches classes for majors and nonmajors, such as general chemistry, organic chemistry, and biochemistry. She received the Ph.D. degree. Her additional areas of interests include students’ professional and personal growth, premedical track advising, extra-curriculum college activities, and combining art and science. Contact her at ppine@luc.edu.

Liudmila Ivanovna Paina is a Docent of Orenburg Medical University and works with students of medical school and residency. In addition to teaching, she is leading a research in various fields of pedagogy. Her general areas of professional interests include pedagogy, phycology, conflictology, and the impact of the sociocultural environment on student education. She received the Ph.D. degree. Contact her at li_pine@mail.ru.
Hitting the Ground Running: Computational Physics Education to Prepare Students for Computational Physics Research

Amy Lisa Graves  
Swarthmore College  
Department of Physics and Astronomy

Adam D. Light  
Colorado College  
Department of Physics

Abstract—Momentum exists in the physics community for integrating computation into the undergraduate curriculum. One of many benefits would be preparation for computational research. Our investigation poses the question of which computational skills might be best learned in the curriculum (prior to research) versus during research. Based on a survey of computational physicists, we present evidence that many relevant skills are developed naturally in a research context while others stand out as best learned in advance.

The benefits of cutting-edge research with undergraduates are undeniable, in computational physics (CP) or any other discipline. Research brings many benefits to students, with special ones attached to students who are early in their career, women and/or members of underrepresented minority groups. The purpose of this paper is to ask how to best prepare a student for CP research (CPR) which in the U.S. typically happens during the summer,
whether on the home campus, or elsewhere via a program like the National Science Foundation Research Experience for Undergraduates (NSF REU). Arriving prepared at a CPR lab, whether at home or away, implies a greater chance of thriving: comfort with the work, higher self-concept as a researcher, faster progress, and more likelihood of eventual coauthorship on conference posters, talks, and refereed papers. Whether in a lab, or as a culminating experience in an undergraduate course or capstone, research is “increasingly important for our students’ careers” (see the Editors’ Introduction).

We ask about preparation as advocates not just for students, but faculty members as well. Faculty at principally undergraduate institutions (PUIs) typically have only undergraduates as their junior partners in research. A track record of cutting-edge research is essential for the careers of most PUI faculty members, making it imperative that there is authentic research progress in an 8- to 10-week summer. The current paper originates in a 2019 talk (found at meetings.aps.org/Meeting/MAR19/Session/F22.3). To our knowledge, we are among the first to publically frame the question of how undergraduate students can “hit the ground running” in CPR.

In 2011, the AAPT recommended that since computation is integral to the modern practice of physics, “every physics and astronomy department provide its majors and potential majors with appropriate instruction in computational physics.” In 2016, the AAPT Undergraduate Curriculum Task Force, produced an informative rationale for fundamental learning of computer-based tools, and modes of computational thinking and doing science. This report identifies challenges departments will face in implementing the suggestions and resources for faculty. Skills are broken into three categories that build upon one another: fundamental, technical, and CP-related. In our study (see the “Survey: What Skills are Needed at What Stage?” section), we attempt to segregate basic, fundamental, and more advanced skills as well as workplace skills like finding information, collaborating with others, and project management (also key in a graduate school setting).

In addition to designing new courses and repurposing existing ones, experts recommend leveraging “opportunities for computational … research projects” and fostering opportunities where “undergraduate research is valued and supported.” In short, CP skills are recognized as the fruit of undergraduate CPR. The premise of this paper is that the reverse is also true. Computational skills are needed to engage in computational research, which builds further skills and advances research progress.

**COMPUTATION IN THE UNDERGRADUATE CURRICULUM**

We make a distinction between CP education (teaching students more physics, more effectively with computers as a vehicle) and education in the discipline of CP. (Utilizing simulations such as Physlets, or PhET simulations from University of Colorado, though excellent pedagogy, does not count as the latter.) Further, there is the well-recognized split between a “scientific computing approach” versus a “computational science” approach to CP education (Chonacky and Winch’s article). Is it better to teach the theory of a $n$th order symplectic integrator; or ask students to find one in a repository on the Web, download and utilize it; understanding its limitations and interpreting the results via the theory of dynamical systems? The “understand everything from the ground up” physicists’ mindset predisposes toward the former. Yet the latter may be better prepared for CPR in dynamical systems theory, and/or the world of employment beyond college.

A decade ago there were only five U.S. undergraduate degree programs in CP (see Landau). An updated report (PICUP and the AIP Statistical Research Center) surveying all U.S. physics departments is forthcoming, but a rough guess can be made that this number has increased but by less than an order of magnitude. Somewhere between 25% and 50% of departments teach computation at the introductory or advanced level. The state-of-the-art roughly a decade ago was ably reviewed in special issues of CISE (Volume 8, 2006) and the American Journal of Physics. Valuable curricular resources in the latter include a
pedagogical resource letter (R.H. Landau) and a guidelines paper based on two years of meetings and surveys with experts and stakeholders (N. Chonacky and D. Winch). That era might be construed as the “2nd generation” of CP education books and courses, in which pure numerical analysis texts gave way to being more oriented toward physics, with support for projects. A clue that CP courses are already structured to benefit CPR is that the faculty responsible for CP curricular innovation have, then and now, been found to be those engaged in CPR.

Difficulties in initiating/sustaining a CP curriculum have been recognized. These “systemic forces” include:

- it’s time consuming (instructors must prepare; time for traditional topics is lost);
- it disrupts the status quo of courses offered;
- it requires new background (for instructor and student);
- it requires more than one faculty member—a community;
- we lack research, from the PER community or elsewhere, on effective teaching of CP.

Hopefully, potential instructors are reassured that they are not alone in encountering these difficulties. They are discussed in light of a recent PICUP-sponsored faculty workshop in the paper by Leary. On the brighter side, there are many resources available for curricular support and change, including:

- campus ITS departments, a number of whom are already offering training in Git, command-line tools, the XSEDE HPC framework, and more.
- software bootcamps, in person or online through organizations like software-carpentry.org, datacamp.com, and Lynda.com
- comPADRE.org, which is a general AAPT-supported materials repository and includes resources specific to CP-like Open Source Physics, and the PICUP project. PICUP, in particular, is well aligned with the goals expressed here and facilitates the “organic” use of computation in traditional courses.
- shodor.org, which features the Journal of Computational Science Education and hosts the Computational Science Education Reference Desk.

Vis a vis preparation for CPR, physics might gain insight from materials science and engineering (MSE). Computational MSE is now an expected departmental research thrust. In 2005, most of the top 20 U.S. MSE departments were planning to introduce computational undergraduate courses and 43% already had, by 2009 the majority of departments had done this. Courses like these tend to teach research-caliber tools; such as industry-standard codes for density functional theory, molecular dynamics, finite element modeling, and thermodynamic phase diagrams; and a notebook environment for calculation/visualization. This is consistent with the fact that most MSE faculties prioritize courses with a computational science approach; for example, only a small minority of MSE faculty wanted students to learn to write code and create new tools. Stated aims of computational MSE courses are, as in physics, a deepened understanding of traditional topics. However, it is clear that MSE courses that teach undergraduates to become confident modelers with state-of-the-art software satisfy the need to hit the ground running in undergraduate research, graduate school, and the workplace.

The PER literature reveals how active learning precepts translate readily to the typical hands-on, team-based teaching of CP in which all stages of manipulating code, data, and results ask students to construct their own knowledge of physics. Since “Most of our knowledge of how to teach ... CP is anecdotal.” and, as mentioned above, few PER studies have concerned CP education, there exist at present “significant opportunity for the PER community to support the development of materials, teaching practices and assessment strategies.” A motivation for the current paper is to explore the opportunity for educational experts and CPR stakeholders to factor in CPR readiness. We wonder which computational skills are best acquired outside of and brought to the lab versus learned (and enhanced) in the lab, and what practices make for inclusive CP education.
SURVEY: WHAT SKILLS ARE NEEDED AT WHAT STAGE?

Respondents who do CP research with students (see the “Forty-Six Computational Physicists Respond to Survey” section) were asked “Please assess the type of experience undergraduate students need before or during their time with you in order to be effective, productive researchers in your computational physics lab.”

Categories A and H below asked for text answers. For B–G, respondents were asked to check one or more of four boxes:

- Ideally, students have prior experience with these
- Students get focused training on these
- Students learn these in the natural course of research with me
- This skill is unnecessary for my work

Survey categories were as follows.

A. Field of Physics in which I do computation-based research.
B. Algorithmic thinking skills.
C. Basic coding skills and knowledge.
   1. Programming in some language at “hello, world!” level.
   2. Calculation/visualization environment (e.g., Mathematica, Jupyter notebook, MATLAB).
   3. Numerical arithmetic (e.g., machine precision, data types).
   4. Flow control (loops, conditionals, etc.).
   5. Scoping rules particular to a programming language.
   6. Selecting and manipulating data structures.
D. Software engineering.
   1. Whiteboarding.
   2. Pseudocoding and diagramming.
   (Note: This answer could be “no”, despite a “yes” to the step of testing code.)

   4. Keeping a TODO list.
   5. Documenting code and keeping documentation current.
   6. camelType and other strong typing.

   7. Interfacing homegrown code with external libraries.
   8. Object-oriented design principles.
   10. Parsing/understanding legacy or inherited code.
   11. Extending functionality of existing code.
   13. Creating modular, well-structured code easily passed on to other students.
   14. Using an IDE.
E. Scientific project and data management.
   1. Keeping a lab notebook or electronic log.
   2. In situ documentation (README files, etc.).
   3. Recording input parameters when codes are run in production mode.
   4. Recording version of code used to produce any set of data.
   5. Naming conventions of output files.
   6. Backing up and knowing how to restore data/codes.
   7. Sharing code effectively with others.
   8. Proactive communication of results and issues.
   9. Collaborative mindset.
F. Fundamental scientific computing skills.
   1. Running an executable code one is given.
   2. Asking people for help with running or writing code.
   3. Reading code documentation.
   4. Googling to answer coding questions.
   5. Locating existing code and libraries on web-based servers.
   6. Terminal/unix shell programming.
   7. Unix shell scripting.
   8. Accessing and utilizing remote platforms (ssh, scp, etc.).
   9. Submitting jobs to a queue.
   10. Using a terminal-based text editor.
   11. Compiling and making.
   12. Reading formatted data.
   13. Writing formatted data.
   14. Understanding compile-time and runtime errors.
   15. Visualizing line data (plots).
   16. Visualizing image data (2-D).
   17. Visualizing volumetric data (3-D).
   18. Creating animations.
G. More advanced computational skills.
2. Profiling code and optimizing execution.
3. Specialized hardware (GPUs, etc.).
4. Using a debugging environment.

H. What have we forgotten to ask about? What do your students need to “hit the ground running” in your computational research?

FORTY-SIX COMPUTATIONAL PHYSICISTS RESPOND TO SURVEY

The survey of the “Survey: What Skills are Needed at What Stage?” section was sent to a total of 105 physicists and astronomy faculty, of which 35 and 70 taught at colleges (PUIs) and universities, respectively. This corresponded to 27 and 29 unique institutions. We began with the U.S. News and World Report 2019 list of 20 top U.S. colleges and universities. Websites for their Physics Departments were scrutinized for computational research that involved undergraduates. Identified faculty members received email solicitations and a link to the survey. In addition, we contacted 19 faculty members known to us for CPR with undergraduates. The overall response rate was 44% (substantially higher for colleges than universities.) The fields of respondents were diverse: atomic physics, astronomy, biophysics, condensed matter, cosmology, fluids, gravitation, materials physics, nonlinear dynamics, nuclear, plasma, soft matter, statistical physics, and quantum: field theory, information theory, mechanics and optics. Compelling questions about differences between PUIs and universities or between different subfields must be left for a future study.

A comprehensive look at the survey data indicates that some skills are better suited to learn “on the job” than others. The scatter plot in Figure 1 captures four dimensions of the data. Each circle corresponds to a labeled skill (see the “Survey: What Skills are Needed at What Stage?” section). We plot skills for which prior knowledge is desired versus those acquired naturally or via focused training in the lab. The circle radius indicates a fraction of respondents for which the skill is needed. The color shows fractions who assert the skill is acquired naturally during research.

A coarse interpretation of this plot is that the skills in the upper left are perhaps necessary to teach in the curriculum. Skills on the lower right can (perhaps should) be learned as needed during the course of the research. The need for basic programming skills [C1] to be developed ahead of time is particularly clear in this representation. A total of 98% of respondents found it necessary, most agreed that students should have prior experience before beginning research, and few believed it develops naturally during the course of research. Of the 52 skills surveyed, six of them had all respondents identify them as necessary:

- [B1] converting a problem to a step-by-step procedure amenable to coding;
- [C4] flow control (loops, conditionals, etc.)
- [E8] Proactive communication of results and issues;
- [F2] asking people for help with running or writing code;
- [F4] googling to answer coding questions;
- [F15] visualizing line data (plots);
- [F16] visualizing image data (2-D).

The type of experience respondents identified as necessary for students to have with these skills was not uniform, however. Responses for [B1] and [E8] above, as well as [C1] (basic programming) and [E4] (code version/output association) are compared in the pie charts of Figure 2. The vast majority of respondents valued these skills, and
they cover a range of the tradeoff between prior knowledge and natural skill development.

An overwhelming number (80%) of respondents emphasized a need for prior experience with programming [C1]. Respondents split fairly evenly on whether the skill [B1] of translating a problem into procedural code was a best learned ahead of time or naturally in the course of research, with another 15% advocating focused training. A third of respondents indicated that students should come in with [E8], ability to proactively communicate results and issues. The mundane but vital skill, valued by 93% of respondents, of matching code versions to data produced [E4] was learned primarily on the job.

DO SURVEY RESULTS MAP TO TYPICAL CP CURRICULA?

It is a rebuttable presumption that a physics course in computation will instill many, but not all, skills ideally needed prior to CPR. For example, consider the following four excellent undergraduate courses:

- Sharma’s freshman-level course for potential STEM majors at Wagner College\(^{15}\) exposes students to work flows in producing, analyzing, and visualizing technical data. With no computer science prerequisites, it distances itself from being a programming course. Mathematica, open-source molecular dynamics, and molecular visualization packages are taught. As with MSE computing courses discussed in “Computation and Careers” section, the emphasis is on using high level platforms for insight into high level questions. Integral to the course are manipulating and visualizing data structures in Mathematica, from molecular mechanics simulation, and from Protein Data Bank files. Some of the substantial final projects have a level of innovative exploration that qualifies as computational research.

- Caballero and Pollock’s PER-informed, Mathematica-based, intermediate classical mechanics course at University of Colorado at Boulder seeks both to deepen an understanding of traditional topics and to prepare students for professional physics research.\(^{14}\) It is pragmatic in terms of student and faculty time. For example, theory of error analysis is replaced with common sense checks like running limiting cases, a “critical part of the modeling cycle” (see Editors’ Introduction\(^2\)).

- Bryn Mawr College has developed an extensive set of materials for both instructors and students.\(^{16}\) Funded by TIDES, a Diversity/
Equity initiative of the AACU and its Project Kaleidoscope (PKAL), a team developed 14 iPython/Jupyter notebook-based modules which embed computational lessons in physics/engineering and social context. Students’ electronic portfolios contain computing work plus profiles of computational scientists and reflections on how what was learned intersects with personal goals. It is clear, for example, from the National Center for Science and Engineering Statistics Women, Minorities, and Persons with Disabilities in Science and Engineering: 2019. Special Report NSF 19-304, that the physical and computer sciences are fields that persistently skewed away from women and people of color. Thus, goals of equity and inclusion are singularly valuable.

- Swarthmore College’s computational lab has undergone many incarnations in 30 years—using Basic, IDL, MATLAB, Mathematica, and (currently) with iPython/Jupyter notebooks. Currently, the course is 7 weeks long and preceded by a recommended workshop, run by the Software Carpentry Foundation of San Francisco, CA and covering terminal editing, Unix shell, and use of a local Github for version control. Topics in numerical analysis are integrated with physical applications, with a final, synthetic exercise on LIGO gravitational wave data.

In Table 1, we have noted which surveyed skills are listed as explicit course goals; with the caveat that names of categories B-G are not rigorous identifiers and are open to interpretation. We invite readers to cross-compare Table 1 with CPR expert priorities displayed in Figures 1 and 2. Also consider the handful of skills which none of the four courses had as an instructional goal: D1, D3, D4, D9, D12; E2-4; F7-9, F11; G1 and G3. These omissions are quite consistent with Figure 1. The least consistent was [D4] keeping a TODO list; even so, roughly 30% of respondents wanted prior experience, around 50% said it would be learned on the job, and 20% it was not needed at all. Understandably, some items are absent because they are outside the technical scope of course tools (e.g., terminal editing using a high level language, HPC). Even so, students did feel the lack of some of these, such as the ability to debug their programs.

Outside the technical reach of many course are certain skills in which roughly 20% of survey respondents wished prior training: [F7] Unix shell scripting, [F8] connecting with remote platforms, [F11] compiling and making, and [E2-4] writing READMEs and keeping track of versions and input parameters. (Item [F6] terminal and Unix basics, was equally desirous for CPR, and a goal of only one of the four courses.) On the other hand, skill [D4] Keeping a TODO list, was one in which roughly 30% of respondents wanted prior preparation. This preprofessional skill worth teaching would not stretch the technical scope of any course.

<table>
<thead>
<tr>
<th></th>
<th>Wagner</th>
<th>Colorado</th>
<th>Bryn Mawr</th>
<th>Swarthmore</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. Algorithmic thinking</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C. Basic coding</td>
<td>1–6</td>
<td>2, 3</td>
<td>1–4, 6</td>
<td>1, 2, 4, 6</td>
</tr>
<tr>
<td>D. Software engineering</td>
<td>6, 11</td>
<td>None</td>
<td>2, 5, 7, 8, 11, 13, 14</td>
<td>5, 7, 10, 11, 13</td>
</tr>
<tr>
<td>E. Scientific programming</td>
<td>1, 5–9</td>
<td>1, 8, 9</td>
<td>1, 9</td>
<td>None</td>
</tr>
<tr>
<td>F. Fundamental computing</td>
<td>1–4, 12, 13, 15–18</td>
<td>1, 2, 4, 15–18</td>
<td>1–6, 10, 12–16</td>
<td>1–4, 12–16</td>
</tr>
<tr>
<td>G. More advanced</td>
<td>None</td>
<td>None</td>
<td>2, 4</td>
<td>None</td>
</tr>
</tbody>
</table>

Notes: i. Absence of D3 is not indicative of a lack of “validation,” since several courses included testing of known cases. ii. D14 may refer to the Jupyter environment, not a separate IDE.

We invite readers to cross-compare Table 1 with CPR expert priorities displayed in Figures 1 and 2. Also consider the handful of skills which none of the four courses had as an instructional goal: D1, D3, D4, D9, D12; E2-4; F7-9, F11; G1 and G3. These omissions are quite consistent with Figure 1. The least consistent was [D4] keeping a TODO list; even so, roughly 30% of respondents wanted prior experience, around 50% said it would be learned on the job, and 20% it was not needed at all. Understandably, some items are absent because they are outside the technical scope of course tools (e.g., terminal editing using a high level language, HPC). Even so, students did feel the lack of some of these, such as the ability to debug their programs.

Outside the technical reach of many course are certain skills in which roughly 20% of survey respondents wished prior training: [F7] Unix shell scripting, [F8] connecting with remote platforms, [F11] compiling and making, and [E2-4] writing READMEs and keeping track of versions and input parameters. (Item [F6] terminal and Unix basics, was equally desirous for CPR, and a goal of only one of the four courses.) On the other hand, skill [D4] Keeping a TODO list, was one in which roughly 30% of respondents wanted prior preparation. This preprofessional skill worth teaching would not stretch the technical scope of any course.

ADVICE AND INNOVATION FROM FOLKS DOING CPR

One CP course that deserves careful mention is focused on CPR readiness and rooted in...
expert advice. Burke and Atherton,\textsuperscript{17} for their 2015 Tufts University course (mostly undergraduates at roughly a sophomore level) modified their initial list of competencies based on interviews with faculty and postdocs active in CPR. Their evidence-based model of expert problem solving was less linear than their original model (see Figure 3), and included meta-competencies seldom found in traditional undergraduate CP courses.

The learning goals, in a nutshell, were: 1) model a physical system, 2) design-implement-validate code, 3) understand the importance of numerical analysis, 4) test hypotheses via visualization and other means, 5) connect theory, experiment and simulation. A physics-content-driven, project-based course is a time-honored way to progressively introduce successively deeper knowledge of the various elements of CP, a fact in evidence in many widely used CP textbooks. (See, for example, Rebbi’s article\textsuperscript{2}; Gould, Tobachnik, and Christian’s \textit{Introduction to Computer Simulation Methods}, 2006; or Landau, Páez, and Bordeianu’s \textit{Computational Physics: Problem Solving with Python}, 2015). It has also been noted that such courses are both consistent with the ethos of a liberal arts education and valuable preparation for sizeable research projects,\textsuperscript{18} to wit CPR, in the future. In the paper by Burke and Atherton,\textsuperscript{17} learning was implemented via short projects in classical, quantum, and statistical physics, culminating in a final project of the student’s choice. A subset of learning objectives were highlighted for each project; these included authentic research-related activities like reading papers, consulting Internet resources, working in groups, running code, testing performance, documenting, and version control. Mathematica and Python were chosen as the required programming languages, but students were allowed to use others for the final project, providing yet another pragmatic lesson: cross-comparing different languages in context. Course evaluations indicated success, but pointed to constructive changes such as additional scaffolded learning for students with different backgrounds, and better management of group dynamics.

Turning to our new survey data, suggestions offered in Item H (see the “Survey: What Skills are Needed at What Stage?” section) can be distilled into various categories. Some respondents I. asserted that students should

- have programming and prerequisite mathematics
- be able to convert a problem to a stepwise procedure for coding (survey Item B1)
- know specific numerical techniques, e.g. FFT, nonlinear curve fitting, or linear algebra
- be able to refactor an existing code
- be able to create new or substantially extend algorithms
- know to test a code, even if it is via a “reality check” via a known case. (Some respondents work in environments which don’t have debugging capabilities.)
- think in common-sense terms about algorithms and workflow. For example

“to structure and automate computations so that the output of one calculation can be efficiently passed as input to the next”
to have algorithmic thinking skills paired with an associated understanding of the underlying science
be proactively communicative
be interested, dedicated, and motivated

II. wondered how to teach their students
be “fearless” about trying what may not work, and persist until they succeed
“soft skills” of computing like commenting/version control/writing READMEs
skills that they themselves may not have learned, or do not employ to the extent that best-practices would dictate. As one said:

“My coding experience was mostly self-taught and I gradually learned the(se) lessons the hard way.”

Several respondents described a self-paced bootcamp experience which they have devised for students. For example:

“I have them log into my computational cluster to work through some standard canned exercises . . . students work in teams. Two students can be working on slightly different versions of the same problem . . . That way, they have a community to talk to, but they have ownership over their own results.”

Finally, while several respondents struggled with the rarity of having students who hit the ground running; at least one (from an R1 university) was OK with this, expecting undergraduates to arrive needing instruction in very basic skills.

A final expert perspective comes from Caballero’s coded notes from interviews with CPR professionals in academia and industry. Caballero, breaks skills into several useful categories, most of which appeared on our survey directly or were addressed in one or more free responses. The only category without representation was meta-knowledge of what a computational solution can deliver. Resonances between the skills in the above-mentioned paper and our survey data are:

- knowledge of certain computational methods, both numerical and data-centered (free responses)
- knowledge of specific physical concepts (free responses)
- move fluidly between math, physics, and computation-related ideas (free responses)
- professional practices like planning/writing pseudocode and debugging (addressed directly)
- data analysis and visualization (addressed directly)
- computational thinking; breaking a complex task into steps which can be encoded (addressed directly)
- meta-knowledge of what a computational solution can deliver (not addressed)
- professional practices like documenting code and version control (addressed directly)

Interviewees also made a point echoed by several of our survey respondents: They were self-taught in professional CP practices. There was agreement, however, that this was not a situation that should carry forward into the education of the next generation of physicists.

CONCLUSION

CP is succinctly defined by the European Institute of Physics (IOP) website as “the science of using computers to assisting in the solution of physical problems, and to further physics research.” The IOP’s Computational Physics Group lists a remarkable array of CP subfields, suggesting it may be hopeless to identify a single type of worthwhile undergraduate preparation for all. Nevertheless, the CPR and PER communities would benefit from coming together to classify types of skills and best teaching practices. This issue is expressed by one survey respondent, who wishes they had an established program to bring students up to a needed level of preparedness “instead of it being a case-by-case, ad hoc training situation,” which works for a small numbers of students, but not in perpetuity with large numbers of students in their research pipeline. Another respondent remarks that one might similarly wish to prepare incoming graduate students with needed breadth of skills. Given their data-intensive field, this person remarks, “you unfortunately have to either teach them computation or physics. As these skills are learned in different schools within universities . . . I’m not aware of any undergraduate program that I think streamlines people to work with me.”
In conclusion, we hope that our current study is a “proof of principle” that experts and stakeholders can teach us which skills are needed, yet not easily mastered, during a brief stint doing undergraduate CPR. In this way, we inform ourselves how to structure CP curricula to help students hit the ground running.

ACKNOWLEDGMENTS

We are grateful to Joan Adler and Rubin Landau for the opportunity to present our work. We thank the colleagues who generously responded to our lengthy survey. Additional thanks to T. Atherton, L. Barba, D. Caballero, C. Crouch, R. Hilborn, P. Jacobs, M. Lerner, M. Matlin, R. Panoff, A. Pattanayak, A. Ruether, A. Sharma, and C. Tunnell. We gratefully acknowledge funding from the Office of the Provost and (AG) a James Michener Faculty Fellowship of Swarthmore College.

REFERENCES


Amy Lisa Graves (formerly Amy Bug) is the Walter Kemp Professor in the Natural Sciences at Swarthmore College. She received a degree in mathematics and physics from Williams College and in physics from M.I.T. Her computational soft matter research has been supported by the American Chemical Society’s Petroleum Research Fund and the National Science Foundation’s Division of Materials Research. She also studies computation in the physics curriculum, and race and gender issues in STEM. She is a Fellow of the American Physical Society, a member of its Committee on the Status of Women in Physics, and on the Editorial Board of the American Journal of Physics. Contact her at abug1@swarthmore.edu.

Adam D. Light is an Assistant Professor in the Department of Physics, Colorado College. He was educated at Case Western Reserve University and The University of Colorado Boulder. His research interests include basic and applied plasma physics. He is a member of the American Physical Society. Contact him at alight@coloradocollege.edu.
IEEE Computer Society Volunteer Service Awards

Nominations accepted throughout the year.

T. Michael Elliott Distinguished Service Certificate
Highest service award in recognition for distinguished service to the IEEE Computer Society at a level of dedication rarely demonstrated. i.e., initiating a Society program or conference, continuing officership, or long-term and active service on Society committees.

Meritorious Service Certificate
Second highest level service certificate for meritorious service to an IEEE Computer Society-sponsored activity. i.e., significant as an editorship, committee, Computer Society officer, or conference general or program chair.

Outstanding Contribution Certificate
Third highest level service certificate for a specific achievement of major value to the IEEE Computer Society, i.e., launching a major conference series, a specific publication, standards and model curricula.

Continuous Service Certificate
Recognize and encourage ongoing involvement of volunteers in IEEE Computer Society programs. The initial certificate may be awarded after three years of continuous service.

Certificate of Appreciation
Areas of contribution would include service with a conference organizing or program committee. May be given to subcommittee members in lieu of a letter of appreciation.

Nominations
Contact us at awards@computer.org
Addressing the Cold-Start Problem Using Data Mining Techniques and Improving Recommender Systems by Cuckoo Algorithm: A Case Study of Facebook

Saman Forouzandeh
University of Applied Science and Technology
Center of Tehran Municipality ICT org

Atae Rezaei Aghdam
University of Applied Science and Technology
Center of Tehran Municipality ICT org

Soran Forouzandeh
University of Applied Science and Technology
Center of Tehran Municipality ICT org

Shuxiang Xu
University of Tasmania

Abstract—One of the most common problems in recommender systems is a “cold-start” problem, which is related to users who do not indicate any behavior on social media. This paper proposes a solution for tackling this problem by using data mining techniques and improving the recommender systems by using the Cuckoo algorithm.

Digital Object Identifier 10.1109/MCSE.2018.2875321
Date of publication 10 October 2018; date of current version 19 June 2020.
**Social Network** is a kind of service that focuses on the online network, which includes different types of people who are interested in sharing their experiences and activities through the Internet. Social network’s users have particular profiles, and these profiles help researchers to acquire more information about users. Finding similarities among users’ profiles is conducted based on various types of techniques, so users’ profiles have become a significant source of information for researchers. Social networks contain complementary information and independent data centers to assist individuals in forming new connections. This type of information plays an essential role in recommendation systems’ performance. Moreover, recommendations to users are based on analyzing their online behaviors and information. Due to the great bulk of information, the process of analyzing data has become a hectic and time-consuming activity. Therefore, data mining techniques can be utilized to discover the hidden knowledge from a bulk of data. In fact, data mining refers to the uncovering new, interesting, and valuable patterns and knowledge from a massive amount of data.

Extracted knowledge from datasets is used in recommendation systems to suggest goods or services to customers based on their preferences and tendencies. Recommendation systems actually try to find a thinking pattern of users, offering the best option to them. One of the most significant challenges in recommendation systems is the “cold-start” problem. This problem occurs due to insufficient information of recommender systems about novel users or new items. In essence, recommender systems cannot recognize a user’s online behavior patterns and also cannot offer the best option to users; thus, it frequently causes poor prediction outcomes for recommender systems.

In this article, we propose a solution for this problem through clustering techniques and association rules. In the next level, for optimizing suggestions to users and increasing the accuracy of results, Cuckoo algorithm has been carried out. The main aim of this research study is for recommending minimum options with maximum acceptance by users. Our case study is the Facebook website, which is one of the most popular online social networks in the world. It includes more than 100 million daily users; each of them log in to their profile at least once a day.

**Literature Review**

Social networks involve complementary information and independent databases. This type of information and the relationship between them can boost the performance of recommendation systems. The purpose of social media is to encourage users to share many attributes with the people close to them. In other words, users are inclined to be touched by the opinions and recommendations from their friends with similar interests rather than from marketing promoters. In this regard, one study analyzed online users’ behavior on the Facebook website to explore the advantages of using it, finding the factors that affect users’ behavior. Due to some characteristics of Facebook, such as those indicating some aspects of individual life (e.g., location, occupation, and email address), people tend to use this social network website more than others. These features led Facebook to become one of the most popular social networks across the globe. The increasing popularity of social network sites helps companies promote their products and services through knowledge discovery and recommendation systems techniques. In addition, customers come across numerous choices to select. In fact, choosing the best option for users based on their preferences has become a hectic debate.

One study analyzed several Twitter profiles and suggested a framework to automatically offer a real-time recommender system, which applied sentiment analysis to explore positive and negative feedback, and achieved a dataset of users with their preferred products without asking...
them to enter ratings and feedbacks. Data were gathered from the Twitter site through Twitter API. Furthermore, clustering techniques and association rules have been used to distinguish and recommend appropriate items to users.11

Another study conducted on Facebook’s social site to propose a social recommender system with a user-oriented approach focusing on relationship between social relations and user interest similarities through sentiment analysis. In the first phase, data were crawled from Facebook website pages, creating a favorite product list of users including cellphone brands, and in the second phase, each user was presented specifications of information, such as year of birth, gender, education level, hometown, interests, etc. The results argued that social features outweigh biography characters in expressing the social relations among users.12 A study done by Forouzandeh and Soltanpanah applied data mining techniques and association rules to propose a recommender system that suggests options of purchase to users based on their behavior on the Facebook website. Data collection performed by Netvizz software, which explores user interests through their likes in other posts and the similarity of user profiles on Facebook and their interests to different products, offered a second choice to users.13

Data mining techniques assist recommendation systems to offer the best options to customers based on their tendencies. For instance, one research study implemented data mining techniques such as visualization, clustering, and association rules to find the best places for tourists based on their favorites. Authors analyzed tourists’ profiles on TripAdvisor, which is one of the popular travel websites. Data were crawled from 600 user profiles through a specific web crawler from TripAdvisor. Then, data mining techniques and qualitative analysis by NVIVO software were carried out to discover the valuable and conclusive outcome. The results of this study were used in recommender system to suggest tourist destination for travelers in Malaysia.14 Another study employed K-means and Cuckoo search optimization algorithms applied to the MovieLens dataset to present a novel recommendation system regarding movie suggestions. They used the well-known Movielens dataset to perform data analysis. Their hybrid model’s performance was more efficient than the current one in terms of mean absolute error (MAE), standard deviation, and root-mean-square error metrics.15

In terms of addressing the “cold-start” problem, there are several research studies that have been published. For example, the research study done by Paolo offered a graph-based and matrix factorization recommendation model to tackle the cold-start problem regarding movie and music recommendation to new users. Datasets are achieved from a structured version of Wikipedia called “DBpedia” and the proposed model was evaluated with a Facebook “like” dataset. Results of this study mentioned that the suggested approach provided relevant recommendation even for users with a very few likes. In addition, the graph-based technique more efficiently elicited content information.16 Moreover, Chou and his colleagues conducted a study in order to address the cold-start problem for next-song recommendation by using a factorization-based algorithm to exploit content features from music and sequential behavior to guess the next song. They studied large-scale music recommendation datasets and found that the proposed algorithm achieved higher performance than other content-based methods. Additionally, their method assists users to explore new options in a next-item recommendation.17 For diminishing the cold-start problem in recommendation systems, Pereira and Hruschka offered a collaborative filtering (CF) recommendation with demographic information. Users are classified into clusters based on their similar characteristics. The method is based on a current algorithm named Simultaneous Co-Clustering and Learning (SCOAL), providing a hybrid recommendation methodology to tackle the cold-start problem. Results indicated that the efficiency of the proposed techniques to replace the lack of information problem for novel users.18
PROPOSED FRAMEWORK

Our suggested framework is depicted in Figure 1.

Phase 1: Gathering Data From Facebook

For collecting data from user profiles on Facebook, we used Netvizz software to obtain a friendship graph for each user. Moreover, Forouzandeh et al.\(^8,13,19\) used the Netvizz application for harvesting data to perform analysis. Data collection includes two sections. Section one consists of user profile information and section two contains an adjacency matrix made by MATLAB software to identify the friendship relations. Thus, the profile data and adjacency matrix were added to the dataset. To analyze user behaviors, we proposed 200 topics related to technology, sport, health, music, and books, asking users to like each item if they were interested. These questionnaires were distributed through five accounts on Facebook containing a total of 3650 friends in which 2780 users fulfilled these questionnaires.

Phase 2: Solving the Cold-Start Problem and Proposing Recommendation System

In the second phase, according to our dataset, data analysis was done in two stages. In the first stage, several data mining techniques including clustering, Density Based Spatial Clustering of Application with Noise (DBSCAN) algorithm, and association rules mining were applied to solve the cold-start problem. In the second stage, the Cuckoo algorithm was utilized to boost recommendation system accuracy in terms of suggesting items to users.

Cuckoo Optimization Algorithm

The Cuckoo algorithm is an optimization algorithm introduced by Yang and Deb.\(^20\) Like other evolutionary algorithms, the Cuckoo optimization algorithm (COA) starts by an initial population of Cuckoos. It was inspired by Cuckoos species. Cuckoos put their eggs in the nest of other birds. Only the eggs that are highly similar to bird eggs have a chance to grow; otherwise, they will be thrown away by the birds. The more eggs survived in the nest, the more benefits we can get. Hence, the location in which more eggs lived will be the best point for us in terms of optimizing parameters.\(^21\) For solving an optimization issue, it is required to convert variables to a form of array. This array is known as “Habitat” in the Cuckoo algorithm. In an \(N\)-dimensional optimization issue, a habitat is an array of \(1\times N_{\text{var}}\) indication which present the living location of Cuckoos. This array is defined as follows:\(^22\)

\[
\text{Habitat} = [x_1, x_2, \ldots, x_{N_{\text{var}}}] .
\]

The habitat profits \((f_p)\) is attained by assessing a profit function based on (2) as follows:

\[
\text{Profit} = f_p(\text{habitat}) = f_p(x_1, x_2, \ldots, x_{N_{\text{var}}}) .
\]

The Cuckoo algorithm maximizes the benefit function. To apply the Cuckoo algorithm in optimization problems, we can use the profit function defined as follows:

\[
\text{Profit} = -\text{Cost (habitat)} = -f_C(x_1, x_2, \ldots, x_{N_{\text{var}}}) .
\]

In the beginning of the process, the habitat matrix is created with \(N_{\text{pop}} \times N_{\text{var}}\) arrays. Then, some eggs are dedicated to each habitat. In
natural life, each Cuckoo lays between 5 to 20 eggs, so that the numbers define a lower bond and an upper bond for the algorithm. They also lay eggs within the maximum distance from their habitats, which is known as the egg laying radius (ELR). In each optimization problem, every variable has an upper bond \( \text{Var}_{hi} \) and a lower bond \( \text{Var}_{lo} \). The ELR formula is presented as follows:

\[
ELR = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{Total number of eggs}} \times (\text{Var}_{hi} - \text{Var}_{lo})
\]

where \( \alpha \) represents an integer, which is supposed to handle the maximum value of ELR.

**METHODODOLOGY**

Our methodology is divided into two phases. In the first phase, we applied density-based clustering and the DBSCAN algorithm, and in the second phase, for improving suggested items to users, the Cuckoo algorithm was employed. Clustering of users was done based on their profiles. In essence, the purpose of clustering in this research was to find users with high similarities with an intended user. The DBSCAN algorithm is one of the density-based data clustering algorithms that offers some concepts such as core point, nearby neighbors, noise, and so on. According to the structure of the Facebook website, selected clusters were based on users’ neighbors, which means that each selected user was put into the cluster with his/her neighbors. In fact, in social media websites, users usually connect to people who are similar to him/her in terms of profile, behavior, or interests. Therefore, the length and number of clusters depend on users’ neighbors. Finally, the items ranked by users were recommended to intended users. For calculating profile similarity, we used the formula defined below:

\[
R'_{c,c'} = \sum_{C \in C} \text{sim}(C, C') \times r'_{c,c'}
\]

where the similarity between user \( C \) and \( C' \) is estimated by \( \text{sim}(C, C') \). The more similarity between users \( C \) and \( C' \), the more accuracy for \( r'_{c,c'} \) in terms of selecting an end user. In the next level, the mutual items among users should be found. For this purpose, we can use cosine similarity or the Pearson correlation coefficient formula. Experimental studies contend that the Pearson formula is more effective than cosine similarity. Thus, the Pearson correlation coefficient formula has been applied in this study. The formula is defined as follows:

\[
\text{Sim}_{i,j} = \frac{\sum m \in (i \cap j)(r_{i,m} - \bar{r}_i)(r_{j,m} - \bar{r}_j)}{\sqrt{\sum m \in (i \cap j)(r_{i,m} - \bar{r}_i)^2} \sqrt{\sum m \in (i \cap j)(r_{j,m} - \bar{r}_j)^2}}
\]

In this formula, \( \text{Sim}_{i,j} \) represents a distance of two items or users. If \( i \) and \( j \) are two users, then \( i \cap j \) represent all items that these two users have ranked them. \( r_i' \) and \( r_j' \) are average ranking of users \( i \) and \( j \) on the same items \( i \cap j \). \( r_{i,m} \) is user \( i \) ranked to item \( m \) and \( r_{j,m} \) is user \( j \) ranked to item \( m \). Due to the fact that user’s ranking in Facebook is a binary, accordingly \( r_{i,m} \) and \( r_{j,m} \) include likes of user \( i \) and \( j \) on item \( m \). Using formula (1) leads to identifying the users that have profile similarity with each other and the intended user. Afterward, formula (2) assists us to compare users’ behavior to extract the mutual items that they liked. Finally, the output can be recommended to users.

In the next phase, the Cuckoo algorithm has been carried out to randomly select a user in each cluster, comparing this profile with other profiles in the cluster. If there is similarity between this user and others, we can conclude that Cuckoo will put eggs in this profile due to the high level of similarity. It should be noted that the main goal of using the Cuckoo algorithm in this study is to find clusters in which we can extend a suggestion to all members of the cluster. Finally, it is assumed that the users that have similar profiles with each other will have similar behaviors, and if they accept one suggestion, their neighbors might accept this suggestion as well.

**ALGORITHM IMPLEMENTATION**

In this research, we applied algorithms and described them in two sections. In the “DBSCAN Algorithm” section, the DBSCAN algorithm is...
carried out, and in the “Cuckoo Optimization Algorithm” section, the COA is implemented. The following sections describe both of the algorithms.

**DBSCAN Algorithm**

To address the cold-start problem, we used data mining techniques. First, the DBSCAN algorithm is used to identify users with profile similarity. This process is done by the following algorithm.\textsuperscript{25}

**Algorithm 1.** DBSCAN algorithm

| Input: D: data set, e: radius, MinPts: minimum number of points |
| Output: \( \pi \): Clustering |
| 1: clusterId = 0 |
| 2: for all unvisited point \( p \in D \) do |
| 3: mark \( p \) as visited |
| 4: \( N = \text{getNeighbors}(p, e) \) |
| 5: if sizeof(\( N \)) < MinPts then |
| 6: mark \( p \) as noise point |
| 7: else |
| 8: clusterId ++ |
| 9: add \( p \) to cluster clusterId |
| 10: for all point \( p' \in N \) do |
| 11: if \( p' \) is not visited then |
| 12: mark \( p' \) as visited |
| 13: \( N' = \text{getNeighbors}(p', e) \) |
| 14: if sizeof(\( N' \)) \( \geq \) MinPts then |
| 15: \( N = N \cup N' \) |
| 16: if \( p' \) does not belong to a cluster then |
| 17: add \( p' \) to cluster clusterId |
| 18: return \( \pi \) |

The DBSCAN algorithm, which is disclosed in Algorithm 1, is an efficient density-based clustering technique that can explore random shape clusters and noise. In the DBSCAN algorithm, the following key phases are executed.

1) Labeling all elements as core, border, or noise points.
2) Putting an edge concerning all core points, which are neighbors.
3) Labeling connected core points as a cluster.
4) Involving all border points within the neighbors of a cluster into the similar cluster.\textsuperscript{25}

The DBSCAN algorithm works based on distance scale, considering the intended user profile as a point with coordination in \((0, 0)\), and for each difference field in neighbor’s profile, the position of the point in axis will be changed; whereas, if the similarity of two profiles are high, the distance between this profile and the intended user will be zero. There were 200 topics regarding sport, health, technology books, and music distributed among users to like their favorite subjects. For applying the clustering algorithm, due to the structure of our survey, which includes five main topics, we choose five users who liked more than others in each subject, and each cluster contains these five users’ neighbors. Given that in this research, the clustering technique is based on user profiles and the DBSCAN algorithm calculates the distance of points, then user profiles formed into binary code as given in Table 1.

According to Table 1, the DBSCAN algorithm was carried out for five users. The outcome is depicted in Figure 2 (it is presented a book cluster results).

Results for five users and final outcomes are presented in Table 2.

**Table 1. Coding user profiles fields on Facebook.**

<table>
<thead>
<tr>
<th>Fields</th>
<th>State 1</th>
<th>Code</th>
<th>State 2</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Under 30</td>
<td>0</td>
<td>Above 30</td>
<td>1</td>
</tr>
<tr>
<td>Gender</td>
<td>Man</td>
<td>0</td>
<td>Woman</td>
<td>1</td>
</tr>
<tr>
<td>Education</td>
<td>Diploma</td>
<td>0</td>
<td>Academic degree</td>
<td>1</td>
</tr>
<tr>
<td>Marital status</td>
<td>Single</td>
<td>0</td>
<td>married</td>
<td>1</td>
</tr>
</tbody>
</table>
As given in Table 2, the highest similarity is related to the technology cluster in which 75% of users have a profile similarity with the intended user. Hence, the technology cluster is one of the most suitable clusters to study user behaviors to extract which item should be suggested to a novel user in terms of solving the cold-start problem.

### Association Rules

The main aim of association rules is to find the most frequent items among users that have high similarity with an intended user containing subjects that gain many likes to recommend to users in the future. The association rules algorithm consists of three elements, which are defined as follows.\(^5,13\)

1. **Support**: It represents a percentage of the total transactions of \(A\) and \(B\) to the total number of records. The higher the percentage of support, the more relation \(A\) and \(B\) have with each other. Support rule is defined as follows:

   \[
   \text{supp}(A \Rightarrow 0) = \frac{P(A \cap 0)}{P(0)} = \frac{P(A \cap 0)}{P(A)P(0)}
   \]

   \[
   \text{supp}(B \Rightarrow 1) = \frac{P(B \cap 1)}{P(1)} = \frac{P(B \cap 1)}{P(B)P(1)}
   \]

2. **Confidence**: It is a ratio of the number of transactions, which include all items in the consequent, as well as the support to the number of transactions. Using this criterion in addition to support can evaluate an outcome of association rules algorithm. The equations below define confidence criterion.

   \[
   \text{Conf}(A \Rightarrow 0) = \frac{P(A|0)}{P(0)}
   \]

   \[
   \text{Conf}(B \Rightarrow 1) = \frac{P(B|1)}{P(1)}
   \]

3. **Lift**: Lift is known as the interest factor, or interestingness, which presents independency between items \(A\) and \(B\). A lift ratio larger than 1.0 implies that the relationship between the antecedent and the consequent is more significant than would be expected if the two sets were independent. Equations (11) and (12) show the lift scale for \(A\) and \(B\), respectively.

   \[
   \text{Lift}(A \Rightarrow 0) = \frac{P(A|0)}{P(0)} = \frac{P(A \cap 0)}{P(A)P(0)}
   \]

   \[
   \text{Lift}(B \Rightarrow 1) = \frac{P(B|1)}{P(1)} = \frac{P(B \cap 1)}{P(B)P(1)}
   \]

The outcomes of association rules techniques are shown as follows.

- **Rule #1**: 129 → 70
  - Support = 0.66159
  - Confidence = 0.93233
  - Lift = 1.2452

- **Rule #2**: 27 → 183
  - Support = 0.63026
  - Confidence = 0.91507
  - Lift = 1.3356

- **Rule #3**: 90 → 160
  - Support = 0.65128
  - Confidence = 0.89437
  - Lift = 1.357

- **Rule #4**: 95 → 117
  - Support = 0.64103
  - Confidence = 0.92616
  - Lift = 1.2757

As it is shown in algorithm outcomes, rules are extracting through association rule techniques and the first rule (above in red) indicate a maximum amount of support. It means that topic numbers 70 and 129 are the most popular topics among users. The second most popular subjects are 95 and 117 in rule 4 with a high dependency ratio. Also, the lift number indicates an attractiveness of these rules. Therefore, extracted items are appropriate options for recommending to users to overcome the cold-start problem.

### Cuckoo Optimization Algorithm

In this section, the COA is utilized to find a high level of similarity between users and the

<table>
<thead>
<tr>
<th>Clusters</th>
<th>User</th>
<th>Number of user</th>
<th>Profile matching</th>
<th>Similarity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health</td>
<td>1</td>
<td>94</td>
<td>52</td>
<td>55%</td>
</tr>
<tr>
<td>Technology</td>
<td>0</td>
<td>217</td>
<td>164</td>
<td>75%</td>
</tr>
<tr>
<td>Book</td>
<td>1</td>
<td>234</td>
<td>98</td>
<td>42%</td>
</tr>
<tr>
<td>Film</td>
<td>0</td>
<td>147</td>
<td>74</td>
<td>50%</td>
</tr>
<tr>
<td>Music</td>
<td>0</td>
<td>119</td>
<td>63</td>
<td>52%</td>
</tr>
</tbody>
</table>

### Table 2. Clusters information.
goal user. First, some nests should be created; then, Cuckoos lay their eggs on these nests. The similar eggs will survive in the nests. In this study, user profiles are considered as a Cuckoo’s egg and the similarity criteria are based on four factors that are given in Table 1. The Cuckoo algorithm is applied according to (13) as follows:

\[
CS = \sum_{i=1}^{4} \sum_{j=1}^{n} \sum_{k=1}^{4} x_{i} \oplus m_{jk},
\]

To make a mathematical model of an algorithm, we define two matrices, comparing them with each other. In other words, we store user profile information in the form of a matrix. This matrix presents as a \( X \), and use \( \sum_{i=1}^{4} \) to read it. As previously mentioned, user profiles contain four fields hence, and we define its change interval from 1 to 4 by variable \( i \). Then, this matrix should be compared with other matrices defined as an \( m_{jk} \) matrix in which \( j \) represents a number of rows and \( k \) indicates the number of columns. Change interval for variable \( j \) depends on the number of eggs (users) in the nest ranging from 1 to \( n \). \( j \) variable represents the number of fields that is supposed to compare with \( X \) matrix. Operator \( \oplus \) is defined to create the output of comparing these matrices. If the amount of two fields is similar to each other, the outcome will be 0; otherwise, it will be 1. The Cuckoo algorithm is defined as follows.

In the first place, some habitats should be provided to put user profiles as eggs into nests for identifying the similarity among eggs (other user profiles). In this research, our case study is the Facebook website and we choose five accounts to study. In each account, we selected two users that have the maximum number of neighbors. Neighbors were put in the habitats to compare with the intended user in terms of profile similarity. Results of the comparison of ten users and ten habitats are given in Table 3.

As it can be seen, each color in Table 3 represents a specific group of users. Moreover, the number of eggs and user profiles are indicated as well as the similarity percentages with other eggs (profiles). According to the results, “Rest 5” is nominated for the best place for Cuckoo eggs because of 73% similarity. Also, the second best place for Cuckoo eggs is “Rest 9” with 66% similarity. Thus, these two habitats are suitable nests for growing Cuckoo eggs. In essence, for users who are located in Rest 5 and Rest 9 habitats, we can recommend similar items due to the high similarity of their profiles with each other. Actually, we can conclude that they will have similar tendency and preferences. For the rest of the eggs, we may choose the most similar profiles (eggs) in every nest and put the eggs in the same nest to find the similarity of profiles among users. In other words, finding users with a high level of similarity in every nest is one of the best approaches to recommend a few items to the vast majority of users. Therefore, we can recommend minimal items to the majority of users that have similar interests.

### ANALYSIS OF RESULTS

In this section, we evaluate the proposed algorithms that are carried out in this research study. Cuckoo algorithm performance is compared with that of the other algorithms. For performance assessment of algorithms, we used some standard measurement parameters consisting of the following.\(^{24}\)

1. **Mean Absolute Error (MAE):** It is a statistical accuracy metric for assessing the average total difference between the anticipated rating and actual rating. A minor MAE value

---

**Algorithm 2. Cuckoo Algorithm**

Input: \( D \): data set, users profile matrix  
Output: \( \mu \): Optimal nest  
1: \( X = \{a; b; m; n\} \)  
2: \( \text{Size} = 1 \) to \( p \): fields of users profile  
3: for \( i = 1 \) to \( \text{Size} \)  
4: \( \text{function} \) compare(\( a \), \( b \), \( m \), \( n \), \( \text{name} \))  
5: \( n = 0; \)  
6: for \( i = 1 \): \( \text{size}(a, 1) \)  
7: for \( j = 1 \): \( \text{size}(b, 2) \)  
8: for \( k = 1 \): \( \text{size}(m, 3) \)  
9: for \( s = 1 \): \( \text{size}(n, 4) \)  
10: if Isequal (\( i(1, a) \), \( j(2, b) \), \( k(3, m) \), \( s(4, n) \))  
11: \( n = n + 1; \)  
12: end  
13: end  
14: end  
15: end  
16: end  
17: \( \text{display}(\text{name}); \)  
18: \( \text{precent}(\text{display}(\text{size}(a, 1) \text{ & size}(b, 2) \text{ & size} (m, 3) \text{ & size}(n, 4)) /100)); \)  
19: \( \text{return} \ \mu \)
indicates more accurate prediction. It is shown in (13) as below:

\[ \text{MAE} = \frac{\sum |r'_{ij} - r_{ij}|}{M} \]  

(14)

where \( M \) is an overall number of predictions. \( r'_{ij} \) represents the forecasted value for user \( i \) and on item \( j \) and \( r_{ij} \) is the true rating.

2. **Accuracy and integrity**: For evaluating an accuracy of model, two measurement factors are used: "Precision" and "Recall." These metrics explore an accuracy of model regarding user behavior prediction. The equations for both precision and recall are defined as follows:

\[ \text{Precision} = \frac{|\text{TP}|}{|\text{TP} + \text{FP}|} \]  

(15)

where TP represents a number of records that systems predicted them correctly. FP presents a number of records that are not chosen in TP domain. Recall is a portion of relevant instances that has been retrieved over total relevant instances. Equation (15) defines recall fraction

\[ \text{Recall} = \frac{|\text{TP}|}{|\text{TP} + \text{FN}|} \]  

(16)

where TP represents a number of records that systems predicted them correctly and FN represents records that are not selected in the whole dataset. Hence, the outcome amount of recall will be lower than precision.

In this article, the main aim of using COA is to find user favorites through minimum recommendation to them, suggesting appropriate items based on their preferences. In this regard, we compare the Cuckoo algorithm with the traditional CF, particle swarm optimization algorithm (PSO), and firefly algorithm (FA) to verify our results. The CF algorithm proposes recommendations to users based on their similarities regarding profiles, ranking items, and so on.\(^{24}\) In this research study, a similarity criterion is based on two factors: gender and education level. In the PSO algorithm, particles tend to move to the best and optimized position in the search-space, which is updated as a better position uncovered by neighbor particles.\(^{26}\) In this study, a user’s profile is considered a particle and other users (particles) are moved to this profile if they have similarity with this user. The FA algorithm is inspired by the flashing behavior of flashing bugs. Randomly produced possible solutions will be measured as fireflies where their brightness is determined by their performance on the objective function. The algorithm works in two phases:

| Table 3. Results of applying COA at ten habitats. |
|---------------------------------|-----------------|-----------------|
| **Cuckoo Egg**                  | **Number of Users** | **% Similarity** |
| Rest 1                         | 0                | 704             | % 52            |
| Rest 2                         | 0                | 593             | % 59            |
| Rest 3                         | 1                | 1389            | % 57            |
| Rest 4                         | 0                | 886             | % 41            |
| Rest 5                         | 1                | 1660            | % 73            |
| Rest 6                         | 1                | 1079            | % 56            |
| Rest 7                         | 0                | 917             | % 47            |
| Rest 8                         | 1                | 914             | % 61            |
| Rest 9                         | 0                | 2398            | % 66            |
| Rest 10                        | 1                | 1705            | % 64            |
updating light intensity and movement. Fireflies are attracted to neighbors with high brightness, and after several repetitive procedures, the optimized solution will come out. For applying FA at our data, each user is defined as a firefly and other users (fireflies) move closer to this user based on their similarities; this is the brightness of goal firefly. Hence, all fireflies that have similar profiles come together, and thus, we can propose specific items to this group of fireflies.

For validating our results of the Cuckoo algorithm, we compared this algorithm outcomes with other algorithms mentioned above. As we choose ten user profiles to study in the previous section, the same profiles have been chosen to analyze by CF, PSO, and FAs. In each algorithm, if users have similar interests with each other, then the accuracy of a recommender system regarding user behavior prediction will be high.

For achieving this goal, we analyzed 300 user behaviors through these algorithms carried out by MATLAB software. We executed CF, PSO, FA, and Cuckoo algorithms to compare the results based on MAE metric, showing the number of users who liked same posts. Results are shown in Figure 3.

Evaluation based on precision and recall metrics are shown in Figures 4 and 5.

As it can be seen, the Cuckoo algorithm is more effective than other algorithms and therefore, this algorithm serves as a breeding ground for enhancing recommendation system performance.

CONCLUSION

Several studies investigated the cold-start problem in developing recommendation systems, but there were almost no research in investigating the combination of data mining techniques and the Cuckoo algorithm to solve this problem. The novelty of this research is about using clustering algorithms, DBSCAN, and association rules along with the Cuckoo algorithm to optimize the recommendations to users. In this research, density-based clustering has been utilized along with the Cuckoo algorithm. Compared to other solutions, our results present a high performance for solving the cold-start problem. We contend that there is some direct relationship between user profile, similarity and their preferences on Facebook. The means that user profile similarities lead to similar requirements and interest. Therefore, they use them for recommending interesting items based on their preferences. In the next level, the Cuckoo algorithm has been applied to find
users with similar behaviors and profiles. Consequently, by offering the minimum number of suggestions, we can meet users’ requirements in a large scale. The outcome of our study can be beneficial for organizations for marketing and advertisement purposes. In the past, organizations had to study user profiles individually to extract their interests. Through the Cuckoo algorithm, they can find user interests and preferences from large groups of users, recommend those items and services, and ultimately reduce advertisement and marketing costs.

**REFERENCES**


Saman Forouzandeh has been a lecturer in information technology and computer science with the Department of Computer Engineering, University of Applied Science and Technology, Tehran, Iran, since 2013. His main research interests include recommender systems, social network analysis, data mining, and Internet of things (IoT). He received the master’s degree in software engineering from the Islamic Azad University, Science and Research Branch, Tehran, Iran, in 2013. He has been selected as an outstanding reviewer for the Journal of Knowledge-Based Systems (Elsevier) in 2017. Contact him at saman.forouzandeh@gmail.com.

Atae Rezaei Aghdam is currently working toward the Ph.D. degree at the School of Information Systems, Queensland University of Technology. He was a lecturer with the Information Technology Department, University of Applied Science and Technology, Iran, from 2013 to 2017. His main research interests include information systems, data mining, social media, and online communities. He received the master’s degree in information technology management from the Universiti Teknologi Malaysia in 2013. He received the Research Training Program Scholarship from the Australian government in 2018 for Ph.D. research. Contact him at atae.rezaeiaghdam@hdr.qut.edu.au.

Soran Forouzandeh is currently a computer programmer (Java, C#, C++) and designer of the website. His main research interests include recommender systems, data mining, and Internet of things (IoT). He received the Bachelor of Software Engineering degree from the University of Orumiyeh, Iran, in 2009. Contact him at soran.forouzandeh81@gmail.com.

Shuxiang Xu is currently a lecturer and Ph.D. student supervisor with the School of Technology, Environments, and Design, University of Tasmania, Australia. His current research interests include the theory and applications of artificial neural networks, genetic algorithms, machine learning, and data mining. He received the Ph.D. degree in computing from the University of Western Sydney, Australia, in 2000. He received an Overseas Postgraduate Research Award from the Australian government in 1996 to conduct his Ph.D. research. Contact him at shuxiang.xu@utas.edu.au.
A Multitier Stacked Ensemble Algorithm for Improving Classification Accuracy

Ramalingam Pari, Maheshwari Sandhya, and Sharmila Sankar
B.S. Abdur Rahman Crescent Institute of Science & Technology

Abstract—For real-world problems, ensemble learning performs better than the individual classifiers. This is true for datasets that have many instances closer to the decision boundary. Using a meta-learner to learn from the predictions of the base classifiers generalizes better. Hence, stacked ensemble (SE) is preferred over other ensemble methods. We extend the SE and propose a multitier stacked ensemble (MTSE) algorithm with three tiers, namely, a base tier, an ensemble tier, and a generalization tier. The base tier uses the traditional classifiers to predict the labels. Tenfold cross-validation is used to validate the models in the base tiers. The cross-validated predictions are combined using combination schemes in the next tier. The predictions from the ensemble tier are generalized using meta-learning to give the final prediction. When tested with 36 datasets, the MTSE gives superior performance over the SE. It achieves high accuracy and does not suffer from over-fitting/under-fitting.

Ensemble learning combines the predictions of multiple weak classifiers to provide the final predictions. Ensemble learning improves the accuracy of classification, the robustness, and the stability of the models. The ensemble performs better when the base models achieve more than 50% of accuracy and are diverse. Hence, the accuracy and diversity of classifiers dictate the effectiveness of ensemble learning. All the accurate models do not disagree with different parts of data, and therefore there is a
need to strike a perfect balance. There are three major reasons why ensemble learning is better than individual classifiers. They are (i) statistical reasons, (ii) computational reasons, and (iii) representational reasons.

Rather than combining the results from the weak classifiers using a static function like average or weighted sum, trainable combiners generalize well. These ensemble systems are called stacked ensemble (SE) or stacked generalization. Here, the combiner is also a classifier, which is trained using the labels predicted by the weak classifiers. The process of learning from the meta-knowledge produced by the weak classifiers is called meta-learning. In these systems, there are two levels of classifiers resulting in two levels of models. They are level-0 models and level-1 generalizer. The level-0 models are validated using leave-one-out $k$-fold cross-validation. The union of predictions for each of these $k$-folds along with the class labels in the original space gives the level-1 input space. Hence, the accuracy of the SE is all about how well it can predict one part of the training set when taught with the rest of the training set. There are two major factors that determine the success of the SE in improving the accuracy. They are (i) the type of features used to form level-1 space and (ii) the classifier used as level-1 generalizer. The class probabilities or the predictions of level-0 models are the most commonly used meta-features. Some of the previous works had used the entropy of class probabilities as meta-feature. Multiple linear regression is the commonly used generalizer, time and again, it has been proved that any suitable classifier specific to the problem on hand can be used.

This study takes the stance that introducing another tier in-between the level-0 models and the level-1 generalizer in the SE leads to better accuracy. The new tier combines the predictions from level-0 models and gets the intermediate predictions, which serve as the meta-features. Using multiple combination schemes in the newly introduced tier leads to a better set of meta-features for the level-1 generalizer. Training the generalizer on these meta-features gives accurate predictions. The contributions of this study are (i) propose the innovative multitier stacked ensemble (MTSE) algorithm to improve the accuracy and (ii) encourage the analytics community to explore this algorithm for their problems. Organization of the remaining sections: The “Related Work” describes the related work in this area of the SE. The methodology and the experimental evaluation of the MTSE are depicted in the “Methodology” and “Experimental Evaluation” sections, respectively. The “Conclusion” section concludes this article and details the scope for future work.

RELATED WORK

The research community has tried out many different extensions of the SE. Seewald et al. proposed Stacking C that used the class probability distribution of the base classifiers as the input for the regression models. He built one regression model for each of the classes in the input space. The output from these regression models was normalized to get the class probability distribution. For each instance of data, the classes with the highest probabilities were the predictions. The labels for the meta-learning were taken as either 1 or 0 depending upon whether that instance belongs to that particular class or not. In the case of binary classification problems, Stacking C used only one linear regression model to predict the class labels. Using linear regression models as a generalizer is not suitable for problems where the conditional variance is not constant.

Developing an SE system to learn the brain images in a hierarchical fashion pays a rich dividend in classifying the brain-related diseases. Liu et al. developed an SE system to diagnose Alzheimer’s disease. A hierarchical ensemble was built to combine the features and the decisions in a gradual manner. From the brain image, \(k\) number of local patches were extracted. For each of these patches, the local imaging features and the correlation-context features were used for training two base classifiers. Thus, \(2 \times k\) number of base classifiers were trained. In addition to the predictions from the base classifiers, the coarse-scale imaging features were used to train the next-level classifiers. The predictions from these classifiers were ensembled using weighted voting to give the final prediction. Training on the features of different brain regions helped in improving the classification. For the brain
regions that were overlapping with each other, a forward greedy search algorithm was used to select the classifiers to maximize the improvement of performance. The training set was divided into tenfold for cross-validation. The frequencies of selection of the classifiers were used as the weights for the voting.

Baumann et al.\textsuperscript{14} proposed a random forest framework that used a cascaded structure with several stages of decision trees to detect the objects. The complexity of the decision trees was increased from one stage to another stage. Weighted voting, full stage rejection, and majority stage rejection along with the bootstrapping were implemented. In the first scheme, the predictions from different stages were combined using the weighted majority voting. The stages with low accuracy were assigned lower weights based on their F1 scores. In the second scheme, the objects were accepted if all the stages passed them. In the third scheme, the images were accepted if more than half of the stages passed them. In every stage, true negatives were removed and refilled with false positives (FP). When tested with pedestrian detection, car detection, and unconstrained face detection datasets, the objects were detected with less time compared to the noncascaded trees.

Ekbal and Saha\textsuperscript{15} used the conditional random field (CRF) and support vector machines (SVM) as the level-0 classifiers. A diverse set of features like context words, word prefix and suffix, word length, infrequent word, part-of-speech information, chunk information, dynamic feature, unknown token feature, word normalization, head nouns, verb trigger, word class feature, informative words, content words in surrounding contexts, and orthographic features were used to train the level-0 models. A genetic algorithm was used to select the features and to optimize the level-0 models. For each of the set of features produced by GA, CRF, and SVM models were trained. The predictions from these models along with the features selected by GA formed the input space for the level-1 generalizer. CRF was used as the level-1 generalizer. Recall, precision, and $F$-measure were used to evaluate the models. When tested on the JNLPBA dataset and GENETAG dataset, the $F$-measure was found to be 75.17\% and 94.70\%, respectively.

**METHODOLOGY**

The major challenge with the SE is that either the bias or the variance of level-0 models gets cascaded to the level-1 generalizer. Due to this, there is still some level of over-fitting or under-fitting. It affects the performance of the ensemble system. To overcome the problem of over-fitting or under-fitting, there is a need to fine-tune the SE so that it achieves better performance. In the literature, many ways of fine-tuning the SE are discussed. Some of them focus on the features used in the level-1 space and others focus on the type of a generalizer used in level-1.\textsuperscript{7} Though these approaches have resulted in improving the performance, they have not reached the desired level of performance. Hence, there is a scope for further improvement in performance. In this study, the SE is extended by introducing another tier between the level-0 classifiers and the level-1 generalizer. The proposed algorithm is named the MTSE.

As depicted in Figure 1, the MTSE has three tiers. The base tier uses a set of weak classifiers, which are suitable for the problem at hand. The ensemble tier uses the combination schemes to produce the combined predictions based on the results of the weak classifiers. The generalization tier does the meta-learning based on the predictions obtained from multiple ensemble classifiers and the class labels in the known samples. This helps to control the bias and the variance in two different tiers. The base tier takes care of reducing the bias, and the other two tiers take care of reducing the variance. This leads to a perfect tradeoff between bias and variance and hence results in the improvement in performance. The base tier in the MTSE uses the following classifiers.

- Support vector machines (SVMs).
- Decision trees (DT).
- Logistic regression (LR).
- K-nearest neighbors (kNNs).
- Gaussian Naïve Bayes (GNB).
- Stochastic gradient descent (SGD).
- Passive aggressive classifier (PAC).
- Perceptron (linear model).
The dataset is split into two sets $S_1$ and $S_2$ at 80% and 20% proportionately. $S_1$ is used for training and $S_2$ is used for the final testing. In the base tier, tenfold cross-validation is used to reduce the bias. The set $S_1$ is partitioned into ten sets of equal size. In each fold, nine of these sets combined are used for training the base models, and the left-out set is used for testing the base models. This is repeated for ten folds by selecting a different set for testing. Thus, each set is used nine times for training and one time for testing. This results in the cross-validated predictions for all the instances in $S_1$. The ensemble tier uses the following combination schemes.

- Plural voting.
- Majority voting.
- Weighted majority voting.
- Confidence-based voting.

Logistic regression is used as the meta-classifier in the generalization tier. The critical features from the input space are selected based on their covariance with the class labels. The top three features with high covariance values are selected. The predictions from the combination schemes along with the selected features from the input space for the generalization tier. Both SE and MTSE are tested with the set $S_2$. Based on this, the accuracies of the models are compared.

The following are the two reasons why the MTSE performs better than the SE.

- Consider the predictions from each of the base classifiers as a random variable. The combination scheme in the ensemble tier is a joint distribution of these random variables. The joint distribution takes all the pair-wise interactions of the random variables into consideration. The multiple combination schemes define multiple joint distributions of the random variables. Each of these random variables is also a function that maps
the input space to the class labels. Due to this, the output of the combination schemes provides features that not only encapsulate the relationship between the predictions from the base tier and the actual labels but also the relationship between the input space and the class labels. These features provide more quality information about the input space than the original features in the input space. Hence, training a model on these features yields models with better accuracy.

- Depending upon the misclassification rates of the base classifiers, the predictions from each of the base classifier contains a set of FP and false negatives (FN). When appropriate classifiers are used in the base tier, all these sets become more or less complement to each other. With a larger pool of base classifiers, the number of common elements between them tends to zero. When the base classifiers disagree for different sets of examples, the combination schemes in the multitier ensure that the weaknesses of the base classifier are neutralized with each other. Hence, the combination schemes produce better features. When these features are trained using a suitable classifier, it generalizes better than the SE.

In addition to the above-stated reasons, the MTSE is suitable for most of the classification problems due to the following reasons also.

- Statistical reason: The ensemble tier in the MTSE combines the predictions of the base classifiers using different combination schemes. This ensures that the risk of choosing a wrong classifier is reduced. The risk is further reduced at the generalization tier, as it generalizes the outputs from multiple combination schemes.
- Computational reason: MTSE completely eliminates the possibility of finding the local optima. The diversity in the MTSE ensures the better approximation of the underlying function between the input data and the class label.
- Representational reason: Even if each of the base classifiers cannot truly represent the underlying function, the combination schemes ensure that the space of the underlying function is expanded. The generalization tier expands this space further. Hence, the MTSE can truly represent the underlying function.

Mathematical Model

Let \( D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \) be the dataset, where \( x_i \in X, y_i \in Y = \{c_1, c_2, \ldots, c_l\} \) and \( N \) is the total number of instances in the dataset and \( c_1, c_2, \ldots, c_l \) are the class labels. Typically, the input space \( X \) consists of many features, and hence its elements are represented as \( N \)-tuple of “\( d \)” dimensions

\[
X = \{(x_{11}, x_{12}, \ldots, x_{1d}), (x_{21}, x_{22}, \ldots, x_{2d}), \ldots, (x_{N1}, x_{N2}, \ldots, x_{Nd})\}
\]

where “\( d \)” is the number of features in the input space. Hence, the dataset is represented as

\[
D = \{(x_{11}, x_{12}, \ldots, x_{1d}, y_1), (x_{21}, x_{22}, \ldots, x_{2d}, y_2), \ldots, (x_{N1}, x_{N2}, \ldots, x_{Nd}, y_N)\}.
\]

Split \( D \) into two sets \( S_1 \) and \( S_2 \) at 80% and 20%, respectively. \( S_1 \) is used as a training set and \( S_2 \) is used as a test set

\[
D = S_1 \cup S_2.
\]

**Base Tier** Let \( WC = \{WC_1, WC_2, \ldots, WC_m\} \) be the set of weak classifiers.

Let \( T_1, T_2, \ldots, T_9 \) be the equal size sets, partitioned from the set \( S_1 \).

Fold 1: The training set \( T = T_1 \cup T_2 \cup \ldots \cup T_9 \).
The testing set \( V = T_{10} \).
Train the weak classifiers using \( T \).
The weak classifiers \( WC_m \) maps the elements of \( T \) into one of the classes in \( Y \)

\[
WC_k : X_g \rightarrow Y
\]

\[
WC_k : X_g \rightarrow \{c_1, c_2, \ldots, c_l\}
\]

where

\[
k = 1, 2, \ldots, m \text{ and } X_g \text{ belongs to } T, \text{ i.e., } X_g \in T.
\]

In the next fold, \( T_9 \) is taken as the testing set and the remaining nine sets combined together are taken as the training set. This is repeated until all
the partitioned sets in $S_1$ are taken as testing sets. This results in the predictions for all the instances in $T$. Let $WL = \{WL_1, WL_2, \ldots, WL_m\}$ be the set of labels predicted by the weak classifiers

\[ i.e., \quad WL_k = WC_k(X_g) \]  

where $k = 1, 2, \ldots, m$.

**Ensemble Tier** Let $EC = \{EC_1, EC_2, \ldots, EC_m\}$ be the set of combination schemes and they combine the labels predicted by the weak classifiers and map them into one of the classes in $Y$

\[ EC_k : WL_k \rightarrow Y \]  

\[ EC_k : WL_k \rightarrow \{c_1, c_2, \ldots, c_l\}. \]  

\[ EC_k = \text{Aggregation of } (WL_k) \]  

\[ EC_k = \text{Aggregation of } (WC_k(X_g)) \]  

where $k = 1, 2, \ldots, m$.

Let $EL = \{Y_{ec1}, Y_{ec2}, \ldots, Y_{ecm}\}$ be the set of labels predicted by the ensemble classifiers. This can be represented in mathematical form as

\[ Y_{eck} = EC_k(WL_k) \]  

\[ Y_{eck} = EC_k(WC_k(X_g)) \]  

where $k = 1, 2, \ldots, m$.

**Generalization Tier** The generalization tier takes the top three critical features $CF = \{CF_1, CF_2, CF_3\}$, ensemble predictions $EL$, and the actual labels $Y$ as the input. It produces the generalized predictions. This can be mathematically represented as

**Input:** $\{CF, EL, Y\}$ or $\{\{CF_1, CF_2, CF_3\}, EL, Y\}$

\[ GL : \{CF, EL\} \rightarrow Y \]  

\[ GL : \{CF, EL\} \rightarrow \{c_1, c_2, \ldots, c_l\} \]  

\[ GL = GC(\{CF, EL\}) \]  

\[ GL = GC(\{CF, EC_k(WL_k)\}) \]  

\[ GL = GC(\{CF, EC_k(WC_k(X_g))\}) \]  

where $k = 1, 2, \ldots, m$, $GC$ is the generalization classifier or meta-learner and $GL$ is the final prediction produced by the meta-learner. $GC$ generalizes the labels predicted by the ensemble classifiers into one of the classes in $Y$.

**Accuracy of Classification** In this study, the accuracy of the classifiers is calculated using the test set $S_2$. The ratio between the count of correctly classified instances to the total number of instances classified gives the accuracy

\[ \text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \]  

where $TP$ is #true positives, $TN$ is #true negatives, $FP$ is #false positives, and $FN$ is #false negatives in the final prediction. As the MTSE focuses on improving the TP and the TN, accuracy is the only performance measure considered in this study.

**Algorithm 1.** Classification by Base Classifiers

1: **Input:** Dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, $x_i \in X, y_i \in \{c_1, c_2, \ldots, c_l\}$

2: **Output:** The labels (WL) predicted by the weak classifiers

3: **Randomly split** $D$ into two sets of size 80% and 20% and name them as training set $S_1$ and test set $S_2$.

4: **Randomly split** $S_1$ into ten sets of equal size 10% and name them as $T_1, T_2, \ldots, T_{10}$

5: **Do for** $m = 1, 2, \ldots, M$: // Repeat for eight weak classifiers

6: **Do for** $i = 1, 2, \ldots, K$: // Repeat for each of the ten folds

7: $T \leftarrow \bigcup_{j=1}^{10} T_j$, except for $i == j$

8: $V \leftarrow T_i$

9: **Train** a weak classifier $WC_m$ with $T$

10: **Run** the weak classifier $WC_m$ with $V$ and get the predictions $WL_{mi}$

11: **End**

12: **Combine** the predictions of all the ten folds of the weak classifier $WC_m$ into a single set. $WL_m = WL_{m1} \cup WL_{m2} \cup \cdots \cup WL_{m10}$

13: **Run** the weak classifier $WC_m$ with $S_2$ and get the predictions $WL_{Tm}$

14: **Calculate** the Accuracy of the weak classifier $WC_m$:

\[ A_m = \frac{(TP_m + TN_m)}{(TP_m + TN_m + FP_m + FN_m)} \]

15: **End**

16: **End**
Algorithm 2. Classification by Combination Schemes

1: **Input:** The labels (WL) predicted by the weak classifiers
2: **Output:** The labels (EL) predicted by the ensemble classifiers
3: **Plural Voting:**
4: Do for \( n = 1, 2, \ldots, N \):
   \[
   EL_n \leftarrow \text{Mode of } (WL_{1n}, WL_{2n}, \ldots, WL_{mn})
   \]
5: **End**
6: **Simple Majority Voting:**
7: Do for \( n = 1 \) to \( N \):
8: Corresponding to each class label, initialize a counter to zero.
9: Do for \( m = 1 \) to \( 8 \)
10: For each occurrence of a class label in the classifier's prediction, increment the corresponding counter
11: **End**
12: **Max_Count** \( \leftarrow \text{Max(All Counters)} \)
13: If Max_Count > 4:
14: \( EL_n \leftarrow \) The class label corresponding to the Max_Count
15: Endif
16: **Confidence-Based Voting:**
17: For all the instances, store the class probabilities of all the classifiers for all the classes into an array
18: Do for \( n = 1, 2, \ldots, N \):
19: **Max Prob** \( \leftarrow \) Max(class probabilities of all the classifiers for all the classes)
20: **EL** \( _n \leftarrow \) The class label corresponding to **Max Prob**
21: **End**
22: **Weighted Majority Voting:**
23: Use the Brute Force Algorithm to find the optimal weights of the base learners. \( W = \{W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8\} \)
24: **WTot** \( \leftarrow W_1 + W_2 + W_3 + W_4 + W_5 + W_6 + W_7 + W_8\)
25: Do for \( n = 1, 2, \ldots, N \):
26: Do for \( k = 1, 2, \ldots, L \): // Repeat for each class label
27: **WSum** \( _{mk} \leftarrow 0 \)
28: Do for \( m = 1, 2, \ldots, M \) // Repeat for each weak classifier // Find the sum of the products of the weights \( (W_m) \) and the class probabilities \( (P_{mk}) \)
29: **WSum** \( _{mk} \leftarrow W_{\text{Sum}}_{mk} + W_m \ast P_{mk} \)
30: **End**
31: **End**
32: **WAvg** \( \leftarrow \frac{W_{\text{Sum}}_{mk}}{W_{\text{Tot}}} \)
33: **EL** \( \leftarrow \) Class label with highest weighted average
34: **Calculate** the accuracy of the generalizer
35: **End**

Algorithm 3. Classification by Meta-learner

1: **Input:** The labels (EL) predicted by the ensemble classifiers
2: **Output:** The final labels (GL) predicted by the generalizer
3: **Calculate** the covariance with a class label for each of the features in the input space // Used for picking the critical features
4: **CF** \( \leftarrow \text{Pick three features with high covariance values} \)
5: **Combine** the critical features and the ensemble predictions
6: **TS** \( \leftarrow CF \cup EL \)
7: **Train** the generalizer with the set TS
8: **Run** the model obtained in step 6 with the set \( S_2 \)
9: **Calculate** the accuracy of the generalizer
10: **End**

Figure 2 depicts the input and the output of each of these algorithms for a simple example with three instances with three features. To keep it simple, this figure only shows how the predictions are made for the unseen samples. Training these algorithms also follows the similar processing. Here, only three classifiers are considered in the base tier, and three combination schemes are considered in the ensemble tier. In this example, only the top two critical features are used for meta-learning. This example demonstrates how the labels predicted by the weak classifiers are combined as a matrix and is used as the input for the ensemble tier. Similarly, it shows how the combined output from the combination schemes and the top two features together are combined as a matrix and used as the input for meta-learning.

Experimental Evaluation

The experiment was implemented using Python with Scikit-learn library. Synthetic minority over-sampling technique (SMOTE) was used for balancing the classes. The experiment was conducted with the datasets listed in Table 1. These datasets are publicly available in the UCI Machine Learning repository. One-vs.-rest and group-vs.-rest strategies were used for grouping the classes and to derive the datasets. Thus, 36 datasets were derived.

For the purpose of benchmarking, the traditional SE, Rotation Forest, AdaBoost, and Extremely Randomized Trees were used. The comparison of the mean accuracy of MTSE and the other four algorithms is depicted in Table 2. MTSE has achieved a significant improvement in
the accuracy for almost all the datasets. Though other algorithms have reached the accuracy of 100% for some of the datasets, the MTSE is not dragging it down, and it also reaches 100% for those datasets. Thyroid-allrep4 is the only dataset for which the performance of the MTSE (99.57%) is little less than that of the SE (99.78%). This is a reduction of 0.21% concerning the performance of the SE and is not a considerable reduction.

The impact of introducing a new tier into the SE was studied across the datasets to find out the number of datasets for which (i) the accuracy improved, (ii) the accuracy remained the same, and (iii) the accuracy went down. This analysis is depicted in Figure 3. Overall, the performance of the MTSE is superior to other algorithms for all the datasets. Amongst the 36 datasets used for experimenting, there are a total of 35 datasets, where the MTSE is either better than other algorithms or on par with them. MTSE has achieved a maximum improvement of 4.59% for Glass2 dataset. Glass1, Allrep23v1, and Ecoli4 take the next three places regarding the improvement in accuracy. For these three datasets, the MTSE has achieved 4.17%, 3%, and 2.89% of improvement over the SE. Only for the Thyroid-allrep4 dataset, the introduction of the new tier has marginally reduced the accuracy. From 99.78%, it has gone a tad down to 99.57%, a reduction of 0.21% in accuracy. This reduction is further analyzed to find the fact that the features in this dataset are not strongly correlated with the class labels. Thus, an important caveat of the MTSE is that it requires the features of the class labels to have a strong

Table 1. Summary of datasets used in the experiment.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Instances</th>
<th># Attributes</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Abalone</td>
<td>4177</td>
<td>8</td>
<td>29</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Shuttle</td>
<td>58000</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Page Blocks</td>
<td>5473</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Heart</td>
<td>270</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Segment</td>
<td>210</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Thyroid-ann</td>
<td>3772</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>Thyroid-allrep4</td>
<td>1947</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 2. Comparison of accuracies of the MTSE and other algorithms.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MTSE</th>
<th>SE</th>
<th>RF</th>
<th>AB</th>
<th>ERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ecoli0v1</td>
<td>99.39</td>
<td>98.61</td>
<td>97.55</td>
<td>98.03</td>
<td>97.96</td>
</tr>
<tr>
<td>Ecoli3</td>
<td>97.22</td>
<td>95.77</td>
<td>93.42</td>
<td>93.66</td>
<td>94.88</td>
</tr>
<tr>
<td>Ecoli4</td>
<td>98.91</td>
<td>96.02</td>
<td>93.97</td>
<td>95.71</td>
<td>95.71</td>
</tr>
<tr>
<td>Ecoli678</td>
<td>100.0</td>
<td>99.39</td>
<td>94.41</td>
<td>98.17</td>
<td>98.17</td>
</tr>
<tr>
<td>Abalone5v19</td>
<td>100.0</td>
<td>99.44</td>
<td>98.29</td>
<td>99.44</td>
<td>99.39</td>
</tr>
<tr>
<td>Abalone7v17</td>
<td>99.11</td>
<td>97.34</td>
<td>97.83</td>
<td>97.34</td>
<td>97.06</td>
</tr>
<tr>
<td>Abalone9v18</td>
<td>95.4</td>
<td>94.78</td>
<td>92.96</td>
<td>93.04</td>
<td>94.27</td>
</tr>
<tr>
<td>Abalone19</td>
<td>98.84</td>
<td>98.6</td>
<td>95.87</td>
<td>95.51</td>
<td>95.58</td>
</tr>
<tr>
<td>Yeast1v3</td>
<td>96.29</td>
<td>94.41</td>
<td>95.69</td>
<td>94.41</td>
<td>95.91</td>
</tr>
<tr>
<td>Yeast0v4</td>
<td>100.0</td>
<td>99.15</td>
<td>99.36</td>
<td>99.15</td>
<td>99.28</td>
</tr>
<tr>
<td>Yeast1v7</td>
<td>96.29</td>
<td>95.81</td>
<td>96.13</td>
<td>92.55</td>
<td>94.76</td>
</tr>
<tr>
<td>Yeast6</td>
<td>98.9</td>
<td>98.34</td>
<td>98.82</td>
<td>96.55</td>
<td>98.19</td>
</tr>
<tr>
<td>Glass2</td>
<td>94.44</td>
<td>89.85</td>
<td>84.33</td>
<td>85.50</td>
<td>88.46</td>
</tr>
<tr>
<td>Glass1</td>
<td>94.44</td>
<td>90.27</td>
<td>86.71</td>
<td>87.50</td>
<td>89.39</td>
</tr>
<tr>
<td>Glass7</td>
<td>100.0</td>
<td>99.02</td>
<td>84.89</td>
<td>99.46</td>
<td>99.52</td>
</tr>
<tr>
<td>Glass5</td>
<td>100.0</td>
<td>99.5</td>
<td>84.68</td>
<td>99.00</td>
<td>98.36</td>
</tr>
<tr>
<td>Glass6</td>
<td>100.0</td>
<td>99.51</td>
<td>84.12</td>
<td>99.00</td>
<td>98.79</td>
</tr>
<tr>
<td>Shuttle5v3</td>
<td>100.0</td>
<td>100.0</td>
<td>98.72</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Shuttle4v2</td>
<td>100.0</td>
<td>100.0</td>
<td>99.01</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Page Blocks2v4</td>
<td>98.78</td>
<td>98.18</td>
<td>98.43</td>
<td>97.88</td>
<td>96.32</td>
</tr>
<tr>
<td>Page Blocks45v3</td>
<td>100.0</td>
<td>100.0</td>
<td>99.89</td>
<td>99.01</td>
<td>99.23</td>
</tr>
<tr>
<td>Page Blocks25v3</td>
<td>100.0</td>
<td>100.0</td>
<td>99.76</td>
<td>99.54</td>
<td>99.54</td>
</tr>
<tr>
<td>Page Blocks1v5</td>
<td>99.34</td>
<td>99.3</td>
<td>99.18</td>
<td>98.81</td>
<td>99.07</td>
</tr>
<tr>
<td>Page Blocks1v3</td>
<td>100.0</td>
<td>99.91</td>
<td>99.47</td>
<td>99.83</td>
<td>99.61</td>
</tr>
<tr>
<td>Heart1v2</td>
<td>100.0</td>
<td>100.0</td>
<td>99.49</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Wine2</td>
<td>100.0</td>
<td>99.03</td>
<td>99.51</td>
<td>98.10</td>
<td>100.0</td>
</tr>
<tr>
<td>Wine1</td>
<td>100.0</td>
<td>99.16</td>
<td>99.38</td>
<td>99.16</td>
<td>100.0</td>
</tr>
<tr>
<td>Wine3</td>
<td>100.0</td>
<td>99.16</td>
<td>99.43</td>
<td>99.16</td>
<td>100.0</td>
</tr>
<tr>
<td>Segment123</td>
<td>92.85</td>
<td>92.45</td>
<td>92.03</td>
<td>88.79</td>
<td>91.96</td>
</tr>
<tr>
<td>Segment1</td>
<td>99.36</td>
<td>99.01</td>
<td>98.65</td>
<td>98.88</td>
<td>98.88</td>
</tr>
<tr>
<td>Thyroid-ann2</td>
<td>100.0</td>
<td>100.0</td>
<td>99.03</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Thyroid-ann1</td>
<td>100.0</td>
<td>100.0</td>
<td>98.97</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Thyroid-allrep23v1</td>
<td>95.00</td>
<td>92.00</td>
<td>94.09</td>
<td>93.12</td>
<td>94.72</td>
</tr>
<tr>
<td>Thyroid-allrep4</td>
<td>99.57</td>
<td>99.78</td>
<td>99.33</td>
<td>99.71</td>
<td>99.71</td>
</tr>
<tr>
<td>Thyroid-allrep3</td>
<td>99.89</td>
<td>99.89</td>
<td>99.29</td>
<td>99.72</td>
<td>99.84</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>97.34</td>
<td>95.57</td>
<td>93.88</td>
<td>94.69</td>
<td>94.69</td>
</tr>
</tbody>
</table>
correlation with the class labels. This caveat sets the direction for the future work for this study. Exploring the use of principal component analysis (PCA) to select the principal components for each of the datasets and then classifying the datasets is an interesting area of research. Using only the principal components rather than using all the features improves both the training time and the prediction time.

Analyzing the performance of the MTSE separately without comparing it with the SE gives useful inferences. Figure 4 depicts this analysis. MTSE has reached the accuracy of 100% for 17 datasets. MTSE’s accuracy is greater than 98% for 28 datasets. MTSE’s accuracy is between 96% and 98% for four datasets and is between 94% and 96% for three datasets. Segmentation123 is the only dataset where the MTSE has achieved less

Figure 3. Impact of introducing a new tier into the SE on accuracy. Due to the introduction of the new tier, for 27 datasets, the accuracy had improved. For eight datasets, there was no change in the accuracy and for one dataset, the accuracy has gone down.
than 94% accuracy. MTSE’s accuracy for this dataset is 92.85%, which is still greater than the accuracies of other algorithms. On the whole, for 35 datasets, MTSE’s accuracy is greater than 94%. Once again this establishes the fact that the MTSE gives superior performance for a wide range of datasets. To understand the rationale behind MTSE’s performance of greater than 98% of accuracy for the 28 datasets, the correlation between the features was studied. It reveals that the features in these 28 datasets have less correlation between themselves. Hence, these datasets have features that are almost independent of each other. On the other hand, the other eight datasets have some of the features that are interrelated. This inference leads the way for future work of this research. Carrying out feature engineering using PCA improves the performance.

MTSE performed well for both balanced and imbalanced datasets. In the case of unbalanced datasets, MTSE’s performance was measured after the dataset was balanced using SMOTE. The performance of the MTSE is marginally better for imbalanced datasets. Hence, using SMOTE for balancing the datasets has a positive impact on the performance of the MTSE. Another reason for the marginal increase in the accuracy is that the number of training examples increases when the datasets are balanced using SMOTE. For this analysis, datasets with an imbalance ratio (IR) of greater than two are considered as imbalanced datasets. Out of the 36 datasets, 31 datasets have IR greater than two, and five datasets have IR less than or equal to two.

There is no significant change in the performance of the MTSE between high- and low-dimensional space. For this analysis, datasets with more than 20 features are considered as high-dimensional space. Out of the 36 datasets, 30 datasets have more than 20 features, and six datasets have less than or equal to 20 features. The dimension of the dataset impacts only the base tier. Beyond which the number of features in the input space is dependent on the number of base classifiers and the number of combination schemes. The performance of the MTSE is much better for a high volume dataset when compared to the low volume datasets. The volume of the dataset impacts all the three tiers and hence a proportionate increase in the accuracy. For this analysis, datasets with more than 50 000 instances are considered as high volume datasets. Out of the 36 datasets, Shuttle5v3 and Shuttle4v2 are the only datasets with 58 000 instances. Hence, there are the two high volume datasets, and the remaining 34 are low volume datasets. For the two high volume datasets, the MTSE has achieved a performance of 100%. For the low volume datasets, the average performance is 98.57%. Another interesting inference is that these two high volume datasets have an IR of 18.62 and 182.38. Due to the high value of IR, SMOTE also has played a vital role in improving the performance of the MTSE for these two datasets.

For the high volume dataset (Shuttle dataset with 58 000 instances), the execution time of MTSE and other algorithms is

- MTSE—17.48 s;
- SE—16.81 s;
- RF—13.53 s;
- AB—12.77 s;
- ERT—11.96 s.

Due to the additional data processing happening in the newly introduced tier, the execution time of the MTSE is higher than the other algorithms. Considering the improvement in accuracy and the computing resources available in the current scenario, the increase in the execution time of the MTSE is not consequential. The parallel processing of the base learners and the parallel processing of the combination schemes in the ensemble tier can be considered for the future work.

CONCLUSION

This study introduced an MTSE algorithm by adding a new tier into the traditional SE. The experimental results point out that the MTSE achieved a better accuracy than the SE. From this study, it is evident that the introduction of a new tier into the SE gives superior performance for balanced/unbalanced datasets, high/low-dimension datasets, and high/low volume datasets. Hence, we recommend the use of the MTSE for the real-time classification problems in any domain. This study can further be enhanced by carrying out feature engineering for the datasets. Feature engineering is expected to improve the performance further and hence make the MTSE
the most reliable solution for classification problems. In addition to this, we are also exploring if the MTSE is suitable for incremental learning using streaming data.

**REFERENCES**


**Ramalingam Pari** is a Research Scholar with the Department of Computer Science and Engineering, B.S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India. His research interests include machine learning, artificial intelligence, and software engineering. He received the M.Tech. degree from Ponnaiah Ramajayam Institute of Science and Technology University, Thanjavur, India. Contact him at pari_ramalingam@yahoo.com.

**Maheshwari Sandhya** is a Professor and the Head of the Department of Computer science and Engineering, B.S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India. She contributed to the research and education in wireless networks, cloud computing, and big data analytics. She is a Life Member of the Indian Society for Technical Education. She is a Reviewer for many international journals. Contact her at sandhya@bsauniv.ac.in.

**Sharmila Sankar** is a Professor of computer science and engineering with B. S. Abdur Rahman Crescent Institute of Science and Technology, Chennai, India. She contributed to the research and education in Internet of Things, wireless networks, and big data analytics. She is a Member of the Association for Computing Machinery and Computer Society of India. She is a Reviewer for many international journals. Contact her at sharmilasankar@bsauniv.ac.in.
Software Development Strategies for High-Energy Physics Simulations Based on Quantum Field Theory

Tomasz Przedzinski
Jagellonian University

Maciej Malawski
AGH University of Science and Technology

Zbigniew Was
Institute of Nuclear Physics

Abstract—Based on typical Monte Carlo simulation projects, we discuss key qualities of scientific software development, including openness to change, robustness against external changes, modular development processes, and the constant evolution of tests. The analyzed projects were developed over more than 30 years, much longer than the typical lifetime of computer industry software. We make general observations together with quantitative assessments on the quality of software, measured by the accuracy and precision of the simulation, together with the complexity of software and tests. Attempts to generalize remain speculative, as every researcher adapts to the specific needs of their own application. In the future, our observations can be validated, adjusted, or extended by experiences gained in other projects.
THE DEVELOPMENT OF scientific software in disciplines such as high energy physics (HEP) is a lengthy process, which may span multiple decades. For the last 50 years, HEP has relied on massive numerical and, in general computing, algorithmic applications for simulation of physics processes. The evolution of software needs not only to match the increasing requirements coming from rapid advances in mathematics, theory, and experiments, but it must also follow advances in information technology and software engineering (SE).

In this article, we analyze the evolution of a family of successful software projects for high-precision simulation of HEP. These projects focus on the generation of physics events using Monte Carlo methods and are crucial for understanding physical processes and simulation of particle physics detectors. We will briefly describe the purpose and history of several projects: TAUOLA\(^1\) Monte Carlo generator for $\tau$ decays; TauSpinner\(^2\) tool for simulating spin effects in $\tau$ decays; PHOTOS\(^3\) Monte Carlo generator for photon emission in decay of particles or resonances; and MC-TESTER\(^4\) tool for a comparison of Monte Carlo predictions. These software projects were successful in the sense that they gained a wide popularity in the HEP community, measured by citations in the literature (see Appendix C, which is available in the IEEE Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/MCSE.2019.2947017).

A crucial aspect of the development of a Monte Carlo tool is how much time and effort has to be allocated on the validation of its results. That is why many of the projects mentioned in this article have separate publications focused solely on the validation process or benchmarks.

The main objective of this article is to quantitatively assess and illustrate how much work (in terms of code and test size) is required to improve the accuracy. The detailed analysis of over 30 years of code evolution based on our repositories shows that there is a correlation between increasing accuracy and precision and the complexity of the code and tests.

The results of our analysis lead us to potentially interesting observations, which may be particularly helpful for developers of projects where effort is required in both software development and studies of the domain. These conjectures for the general aspects of work on scientific software include robustness against external changes, modular development process, constant evolution of tests, openness to change of the design, and looking ahead for future extensions and applications.

RELATED WORK

The accuracy and precision of scientific software has been the subject of research by software engineers. In one of the early studies,\(^5\) the authors investigated several software packages used for seismic data analysis and discovered disturbing nonrandom numerical disagreement between the results of different packages. They estimated the growth of this disagreement to be around 1% per 4000 lines of FORTRAN code. The authors recommend using these cross-testing results to provide feedback to software developers in order to correct software errors.

In this article, we present a different perspective: we show how much work (in terms of code and test size) is required to improve the accuracy.

Development of scientific software has been studied by researchers in the SE domain. Wilson et al.\(^6\) noted that at least 90% of scientists are self-taught when it comes to software development and lack knowledge of basic software development practices. A similar observation was made by Baxter et al.,\(^7\) who also noted that the awareness of tools and good practices was growing. Both works are a good source of simple best practices that could greatly benefit scientific software developers. Additionally, the paper by Kelly et al.\(^8\) lists five techniques that are specifically useful to computational scientists,
namely code separation, organization, review, testing, and simplification.

Sanders and Kelly\textsuperscript{9} emphasized the need for testing not only the correctness of software in terms of agreement with theory, but also the implementation, when the project gains a wider user base. These ideas have been further developed into more formal models of scientific software development.\textsuperscript{10} In this article, we confirm that tests and separation of interfaces are important, and moreover, we provide quantitative evidence of how the size of the code increases when we improve the accuracy and precision.

Goble\textsuperscript{11} noted that software plays a vital role in scientific research, and yet its role is undervalued by the scientific community, despite many instances in which important publications were retracted due to software bugs. She also states an important fact that we have found to be true in our case as well—scientists should not be trained to be computer scientists; instead, best practices for software development should be tailored to meet scientists’ needs.

An example solution for detecting software numerical instability was presented in the paper by Tang et al.,\textsuperscript{12} combining stochastic and infinite-precision testing. Similar techniques, or at least as feasible as it was possible in the earlier years, were used in our KORALW and BHLUMI projects, where quadruple precision calculations were used to verify the numerical stability of the algorithms. In TAUOLA, on the other hand, analytic computations with 12 digits of precision were used to validate the results.

The literature on the testing of scientific software\textsuperscript{13} lists a number of issues unique to this type of project. We have found these descriptions to be quite accurate when related to software for the physics community. The authors stated, “None of the primary studies reported the complexity of the software in terms of measurable units such as coupling, cohesion, or cyclomatic complexity.” This is connected to the fact that the SE techniques known from the software industry are often less efficient in scientific environments. Code metrics such as these do not reliably show parts of the code that could be simplified or refactored.

Sanders and Kelly\textsuperscript{9} noted an important factor that influences the software development process in the scientific community: more than 25% of the code scientists write is written only for themselves; a total of more than 50% of the code is aimed at a group of less than ten people. This striking statistic shows that the usual rules aimed at building code that can be uniformly understood, modified, and tested by a large community do not apply here. It is enough if a handful of experts will be able to efficiently work with the code. That is the main reason why the mathematical and physical model and key elements of the numerical algorithms are commented upon in great detail, some even worthy of their own publications, while the structure of the project is mostly presented “as is” in hopes that its implementation is readable enough, at least in comparison to other complexities present in the project.

Although many approaches are available for testing scientific software (see Kanewala and Bieman,\textsuperscript{13} for example), there have been no studies on the relation between the size of the tests and the accuracy and/or precision.

**PHYSICS BACKGROUND**

In this section, we present the most basic concepts of physical analysis, focusing on the most common type of HEP simulation: a Monte Carlo simulation of the effects of a collision of two particles.

In HEP terminology, such a collision is referred to as “an event” and its data is stored in an event record. An event record is a data structure describing all details regarding the starting point and all of the products of the collision. For simplicity, we will refer to them as “particles.” The most basic properties of a particle are its four-momenta, its mass (referred to as “virtuality” for some particles), its identifier, and its status (see the glossary, Appendix E in the supplementary material, available online). A particle (often referred to as a “mother”) can decay into other particles (“daughters”). A particle can have more than one mother. It is common to record the initial collision as two particles annihilating (merging) into an
intermediate boson, which then decays into other particles. A particle is considered stable (or “final”) when it has no decay products.

It is worth noting that, for the sake of describing some unique phenomena, exceptions are introduced to this data structure. They can, for example, introduce cycles within the tree structure, effectively transforming it into a graph. Quantum interferences, and in particular, quantum entanglements, force developers into sometimes awkward extensions.

A simulation can be thought of as a chain of tools that produce, modify, or analyze the content of an event record. This description immediately points out the three types of tools available for physicists.

- Generators: Calculate products of the initial collision or products of a decay of a particle within the decay tree. For example, a TAUOLA tool described in “Tauola” is a generator that produces decay products of τ leptons. The distributions of the four-momenta and mass (virtuality) of the generated particles are, in general, calculated using complex matrix elements.

- Postprocessing tools (sometimes called “afterburners”): Modify the decay tree with respect to its predefined role. For example, a PHOTOS tool (see the “Photos” section) is used to modify the decay tree by adding additional photons.

- Analysis tools: Do not modify the event record, but extract information from it. For example, MC-TESTER (see “MC-TESTER”), creates a set of histograms based on stable decay products (particles with status 1) of all particles with a given identifier found in a series of events.

An analysis chain can contain any number of tools of any type and in various orders. An example of several tools interacting with each other through an event record is shown in Figure 1.

The tools are usually designed to work independently, and they are often provided by different authors. As a consequence, a tool has no way of knowing other tools preceding or succeeding it. The tools have to inspect the event record carefully for required information and not expect it to always be complete or valid.

This is the most common source of problems that authors of such tools must face. If the tool modified an event record, it must strive to produce a correct event record at any time and minimize potential destructive modifications. Required input and possible results of its use should be well documented as one incorrectly used tool may damage the whole analysis.

An analysis typically must process a large set of events in order to obtain meaningful results. An example of such results is the histogram describing the distribution of the mass or the couplings of particles into which τ lepton decays. The result of the analysis and its statistical error depends on the size of the data sample.

Program, Its Boundaries, and Interfaces

Kelly pointed out that software written by the scientific community is, in most cases, targeted at end users who not only use software tools, but expand them or build new tools on top of them. Therefore, it is important to define interfaces to the program, which means the code responsible for interacting between the programs and code used for extending the program. This interface should not be confused with the term application programming interface (API) used in software development. Scientific programs do have an API, but we will not discuss or refer to it in this article.

An interface of the tool can include not only the data import/export procedures for a specific event record format, but also important
algorithms for processing complex structures in the event record. It also plays an important role in documenting the code and preparing specification for future extensions. As a consequence, the software tool contains not only knowledge related to the problem at hand, but a description of its potential extensions as well. The amount of work spent on the future interfaces is one of the aspects of scientific code development missing from the papers referenced as related works (see the “Related Work” section). It is natural for scientists to document both existing analysis as well as its possible followups. Very often, when working on the related code, appropriate comments are also placed. Some of these comments may not be precise, but nonetheless have great potential value. They document intellectual effort performed during the development.

An interface of a scientific application exposes its inner algorithms and structures, describing how these algorithms can be used and how to modify them. In contrast to API, such an interface is not meant to be immutable. With each new development step, a previously existing interface is refactored and refined as the developer gains an understanding of which parts of these interfaces are essential only when reaching the new step.

Development of interfaces is an important and crucial work and requires much effort in terms of lines of code and tests.

Evolution of Algorithms Driven by Accuracy and Precision Demands

Physics accuracy can be thought of as a comparison of different levels of approximation of a theoretical model. For example, if perturbation expansion $\sigma^{(n)}$ is used, then the next accuracy threshold would be the next expansion $\sigma^{(n+1)}$, exactly as taking into account one more term in the Taylor expansion. In such setup, the statistical error of a difference $\sigma^{(n+1)} - \sigma^{(n)}$ is $\frac{1}{\sqrt{N}}$, where $N$ is the number of samples generated during the simulation, but usually more refined techniques are used to compute it. This error should be at least three times smaller than the error that comes from the physics model to be considered insignificant. The same goes for technical precision. Assuming normal distributions, if technical uncertainty is three times smaller than statistical error, it contributes to the model error at the second significant digit, which is a reasonably small contribution. As technical precision, we understand all practical consequences resulting from numerical calculations. We have in mind the length of a computer word, accuracy of numerical integration, numerical issues of the implementations, etc.

Increasing precision in high energy accelerator experiments is intimately related with increasing statistics of a collected data sample. Results of the measurements are of stochastic nature. This means that increased statistics give the benefit of lower statistical error, and that of looking onto the measured property with greater detail as the sensitivity of different properties of the model grows with an increased data sample. For example, a 3-D property measured with low statistics can be reasonably analyzed only when projected onto a lower number of dimensions. As such, we had to compare our model to several 1-D data distributions.

However, when more data became accessible later on, more detailed models could be considered. In our example, we were able to create 2-D data distributions. It turned out that our model was completely off when compared to a 2-D model.

It is important to note that these future steps have to be taken into account as early as possible to avoid, or at least to minimize, the effort of adapting the model to future analyses, not only in program design, but also in tests.

TESTING

Testing is a crucial step of project development. This is especially true in the physics community where tests are used not only to verify technical aspects of the software, but mostly to validate physical assumptions put into the code. Here, we briefly list the most common types of tests performed for physics projects. The growing complexity of implemented solutions also requires more effort to be put into the development of these tests.

Numerical Technical Tests

Technical tests are by far the most important, but also the easiest to write, automate, and
update. They played the most crucial role back when compilers could not be trusted to always follow standards and produce correct output. A set of numerical technical tests could very well produce different output in different user environments. These differences were either negligible, related to compiler optimization, or catastrophic to the point where the output showed only NaNs. To address these inconsistencies, the Large Hadron Collider (LHC) Computing Grid (LCG) project provided users with a compiled and validated set of libraries for each platform, to be used as a source of software distributions of a tool.

The need for such validation remains, if only to catch sudden changes in compiler behavior. However, with a constantly expanding list of supported platforms and compilers, this process became inefficient, and ultimately even the LCG project stopped the activity of releasing compiled libraries, leaving the responsibility of performing technical tests to the user and that of providing such tests to the developers. Conceptually, these tests are closest to configuration tests with their main goal being to test deployment setup.

Tests of Data Flow

Before event records became widely used, Monte Carlo tools were tested by preparing a separate program for each subcase handled by the tool. There was no need to test a variety of possible inputs. This, however, became a necessary step when input for the Monte Carlo generator started to be produced by a variety of different tools and in various formats. As a result, invalid or incomplete data could have easily been provided to the tool. A tool had to be able to distinguish between invalid cases to avoid problems when used by mistake. After all, it was not easy to set up the whole generation chain correctly.

This mostly concerned algorithms used to traverse the event record. An interface to event records had to be able to handle all exceptional cases and divergence from the standards used by the scientific community, or to issue a warning. Conceptually, these tests are very similar to integration tests with one distinction: that the tested integration is between the developed tool and a set of external tools.

Tests of Functionality

This category contains tests that are most important and hardest to write. Results of the Monte Carlo tools are validated through comparison with other tools, other configuration options, and through comparing the same tool with different level of physics assumptions, or with different implementations (for example, semianalytical integration instead of the Monte Carlo simulation).

By comparing results of the same tool produced using two different levels of precision, verification of the higher precision computation can be performed, regardless of the initial conditions on which the tools are running. In fact, it helps verify that these calculations work correctly in all tested initial conditions. Conceptually, these tests can be thought of as compliance tests; verification that the implemented model matches the theory behind it and produces expected results.

Useful Testing Techniques

As pointed out by several papers mentioned in the section “Related Work,” one of the key problems in testing scientific code is the lack of an “oracle” that could reliably verify the results of a test. That is why a “fuzzy” measurement, such as code’s trustworthiness, is used instead of the binary value determining whether or not the code works correctly. However, to measure the trustworthiness of the code, its results must still be compared against some reliable benchmark; be it an output of another tool, results of a previously conducted experiment, or something else entirely. There are two techniques that we have successfully used in our projects that are similar to the ones presented in related work, but adapted to physics software.

Probing Results at Different Steps of the Computations

One very useful property of the Monte Carlo simulation is that the computation process can be tested at its different steps. Such partial computations still produce meaningful results and, in some cases, they can be compared to something meaningful as well. For example, the simplest step of a phase–space generation is the constant matrix element or narrow width approximation of intermediate
resonances. For this distribution, a simple analytic formula exists. With high enough statistics, we can easily check if the Monte Carlo simulation can reproduce such a formula.

The existence of such “benchmark points” is what drives developers to construct configuration options that allow switching on and off various parts of the Monte Carlo generators. Turning enough options off may allow users to compare the results of the generator to a bare-bones model of their own, or results of an older analysis that did not take into account such new options.

There are also more benefits of introducing such options. Knowing well what kind of effect a particular option introduces, physicists can validate if this option works correctly by comparing results with and without this option. This is as close as one can get to unit-testing a Monte Carlo generator.

A good example of such tests are analytic tests of TAUOLA. Benchmarks used for these tests are of infinite precision.

Comparing Model Results to Semianalytic Calculations It is an extension to the method described above. It is not viable at higher levels of complexity of the model, and at some point not possible, to produce analytic expressions for comparison. However, other means may exist.

One basis of the article on the RChL project (Appendix A in the supplementary material, available online) was a set of semianalytic distributions to which we could compare our results. As described in Przedzinski et al.,14 an effort spent on analyzing the potential of semianalytic distributions allowed to speed up the calculations by several orders and greatly simplified distributed computations.

It is worth noting that such comparisons are usually built on top of an old, currently unused code. Inspecting a tool such as TAUOLA reveals that a lot of unused decay channels left in the code are exactly such benchmarks. They do not always provide physical results, but are suited to test parts of the generator, or for technical precision tests that should be rechecked every time a new precision level is reached.

The ability to cross-check parts of the project with results computed by different means is very helpful in building up confidence in the code.

SHORT HISTORY OF THE TOOLS

Let us briefly recall the histories of several Monte Carlo simulation tools, with the focus on software development aspects. In all of these cases [see Figure 2(a) in the supplementary material, available online], the physics precision and applicability domain were essential driving forces. Below, for each project, we try to find key decisions made during their development in an attempt to recognize which factors contributed to their success or failure.

KORALB

The first program of our team member was published in 1984. It was a starting point for other projects and was used for simulating $e^+e^-$ collisions and the production of lepton pairs. The center of mass energies of the generated events were below 45 GeV. For higher energies, the KORALZ tool, and later the KKMC tool, were developed. Many prototypes for the solution were presented in this one and used later in other projects. Microalgorithms for $\tau$ decays into leptonic and $\pi^\pm\nu_\tau$, $\rho^\pm\nu_\tau$ channels were developed within this project. It also presented the first distributions of benchmarks in $\tau$ decays. Notably, a complete theoretical basis for the algorithm calculating spin correlations, later used in TAUOLA and TauSpinner, was proposed and analyzed during the work on this project.

TAUOLA

TAUOLA was an offspring of the KORALB and KORALZ projects. It began as a small module for the KORALB Monte Carlo generator to produce decays. Unexpectedly, it became a widely used tool. At the time (around the year 2000 and earlier), technical model precision was at 1%. At best, 100k events could have been generated. HEPEVT was the most common event record. In fact, it was the first event record format used widely in the HEP community.

In the beginning, only tests of the standalone generator used small tools HBOOK and glibk to plot 1-D distributions. Semianalytic and analytic reference results were also exploited. The program was used by all LEP experiments and CLEO.

The year 2001 witnessed the introduction of technical precision of 3 permille. TAUOLA was used and modified by new experiments—BaBar and Belle. To handle building different variants
of TAUOLA suited for these experiments, a software solution imported from CLEO collaboration was used and a “factory” for generating the TAUOLA code from different parts of the code was used where FORTRAN code of the tool was preprocessed using cpp. At this point, the tests had to substitute the single precision floating point arithmetic by double precision. Essentially, a large number of significant digits of the result were lost when performing many additions; a trivial observation, but such mistakes are difficult to find if attention is concentrated elsewhere. At this step, TAUOLA Interface started to appear as an independent project.

More recently, when a C++ HepMC event record gained a larger user base, it quickly became obvious that it would be beneficial if TAUOLA could use it. A C++ interface to TAUOLA was created, keeping the legacy code as a standalone black-box module called TAUOLA-FORTRAN. Thanks to that separation, the project could benefit from the input of the low-energy physics community (e.g., CLEO, BaBar). This allowed the original module to be extended and modified by a large user community while allowing access to the newer community that used C++ event records. Different TAUOLA-FORTRAN variants could have easily been used without modifying C++ interface. This interface created an abstract layer to any C++ event record.

The evolution of HepMC created many new use cases and also new problems (see the “Physics Background” section). A great deal of development effort went into adapting the interface itself, and many technical tests had to be added. Complete separation of the interface and main code made it easy to test new changes to the interface. A new interface also allowed for easier addition of new modules, such as SANC electroweak corrections.

TAUOLA is notable as a starting point for several other projects, as shown in Figure 2(a) in the supplementary materials, available online. It is not uncommon for a new project to start as a followup on a work on an analysis. Similarly, test programs created to validate a single Monte Carlo generator can then become useful as standalone tools themselves.

This project shows how openness to change can lead to potential new projects. If your project spans decades, do not block the potential for changes to be introduced in the future. In fact, take into account that any part of your project may require modification. This is one of the reasons why much of such code does not introduce any complex structures or abstract internal layers. Modification of such layers would trigger a cascade of changes.

Development of such software requires compromising between attaining clear software design and allowing the scientist to develop algorithms in a flexible way. The initial design of the project may need to be redrawn and some modules may not survive the changes introduced with time. Try to anticipate such changes ahead of time, even if they are relying on unconfirmed, not fully formulated possibilities for variants. Consider the following, for example.

- Physics and physics community: What are the current approximations of the model? How can they potentially be improved in the future? What are possible future demands for the models? What kind of new data can we expect from the experiments?

- Mathematical input: How can new models, algorithms, or methods help? Which of them are applicable or can be made applicable in the future?

- SE: What is the trend in the community in the use of data analysis frameworks, toolchains, event records, or programming languages? How can the usability of the tool be improved? To what frameworks could, or should, it be adapted? Consider, for example, that the growing popularity of Python can increase a potential user base and compatibility with modern data analysis frameworks.

Taking these factors into consideration early in project development may greatly reduce the effort of extending the project in the future.

**TauSpinner**

At some point during TAUOLA development, an algorithm to compare two different models was needed. A module called tau_reweight has
been created. It allowed applying weights (see glossary, Appendix E in the supplementary material, available online) to an event calculated as a ratio of matrix element of one model to the other. This allowed the comparison of two different models without the need to generate a new data sample. This reduced the statistical error for the model difference only as both models used the same event sample. This algorithm was the basis of the TauSpinner algorithm, which uses weights to add or remove spin effects to tau decays.

The modular approach to TauSpinner created a flexible tool. It could use any version of TAUOLA as a separate library. It applied weights to an event taken from any event record. TauSpinner has been extended to allow generation of arbitrary weights, including those generated by models outside the standard model.

Unexpectedly, TauSpinner became very popular (see Figure 4 in the supplementary material, available online). It allowed the application of the spin effect on already generated samples, saving computing time. It also allowed for quick testing and comparison of results for different variants of a particular model or different models. Because of this, a wide range of use cases were prepared and several papers were produced that document these applications. TauSpinner has become our fastest growing tool in terms of possible application. Considering that it started off as a simple reweighting algorithm for TAUOLA, this outcome was hard to anticipate.

While code itself cannot always be divided into well-established modules, the development process can. The team can often approach mathematics, physics, and computer science aspects of the project separately. However, the team always has to maintain the flow of information between these domains. Otherwise, the work produced by some team members may be unusable by others and the work of the team may become ineffective. An example of such outcome is the RChL project (see Appendix A in the supplementary material, available online). At one point, the effort put into fitting the model to the data greatly surpassed the effort put into building it and in the software design itself, which consequently stalled the whole project.

PHOTOS

Before the time of the PHOTOS publication (1990), precision of data was at percent level. Accurate theoretical simulations, in particular photon emission in decays of resonances, were not needed. The nature of dynamics for physics processes lead nonetheless to final states where additional photons (bremsstrahlung photons) are present. In approximation, such additional photons are described by simple and universal dynamics, so-called eikonal factorized terms. In general, every event is accompanied by an infinite number of small-energy photons. Those photons, up to a certain precision level, can be ignored. Once precision improves, more of those photons needs to be taken into account.

When precision required by experiments improved to 3 permille, it drove the demand for a higher precision solution. PHOTOS started off as a part of TAUOLA; a Fortran routine named RADCOR. Only single photon emission was available. Tests were performed using only a tool for plotting histograms, glibk, and a taustest program originally written for testing TAUOLA.

An algorithm for two-photon emission was added soon after. This algorithm was quite successful (see C) and the program became widely used despite severe problems due to rounding errors. The algorithm was iterative and its consecutive application led to the accumulation of rounding errors. At this time, the first attempt to translate the algorithm to C++ was started. It was bound to have limited applicability because no C++ event record had yet been established in the community. However, at that time, the development of FORTRAN compilers slowed down, so there was a risk that they may disappear. As a consequence, formal analysis of the code was performed. Although this side project failed, it provided essential tests for future development of the project.

When precision improved to 1 permille, PHOTOS became a widespread tool (see Figure 5 in the supplementary material, available online). Due to precision demands, its development has become more important than the development of TAUOLA. It has been extracted from TAUOLA as a separate generator. Soon after, an algorithm for four-momentum rounding error corrections was added, which was a prerequisite for extending the algorithm to produce more photons.
Later, when precision improved to less than 1 permille, the focus of PHOTOS development shifted almost entirely on testing and validation. MC-TESTER, originating from tests of TAUOLA, was used to automate tests at next-to-leading order (NLO) precision (see glossary, Appendix E in the supplementary material, available online). While PHOTOS provided NLO precision, tests were performed with next-to-next-to-leading order reference distributions obtained from KKMC.

PHOTOS followed TAUOLA in its adaptation to C++ interface to event records. Around 2009, data structures of PHOTOS were modified to include information about particles’ grandmothers. This allowed for the introduction of matrix elements, which were not developed in FORTRAN because such precision was not required. Thanks to this, a framework for prototyping new applications became available. Later, it was extended with an algorithm for pairs emission (Appendix E in the supplementary material, available online) and new extensions are currently being worked on.

PHOTOS is a good example of how project evolution is driven by the HEP community. As long as potential users appear, new functionality, extensions, or even new projects can start.

MC-TESTER

MC-TESTER started as a satellite project of an attempt to translate PHOTOS to C++. On the advent of C++ event record era tautest, working only on a FORTRAN HEPEVT event record was quickly becoming obsolete and had to be rewritten.

MC-TESTER was written only using C++, which at that time was a paradigm shift in HEP environments. It was built to handle any type of event record and used an abstract layer of its own classes to represent event structure. This project survived for quite a long time because by design it was able to perform crucial analyses despite some limitations of both the HEPEVT and HepMC event records.

The first version of MC-TESTER used ROOT data analysis framework as its basis for creating histograms and storing data. This approach had its risk. It was too early to tell if ROOT would remain to be used in HEP. Also, at this stage ROOT was the subject of rapid changes (including APIs), which had to be followed by MC-TESTER. Furthermore, external dependency on specific ROOT versions MC-TESTER was compiled with severely limited use of MC-TESTER as a software distributed for LHC.

To deal with this dependency MC-TESTER has been updated so that it could be built without ROOT; an idea that should have been pursued with more effort, but was handled incorrectly, predominantly because of manpower limitations. The usability of this option was very narrow and with time no longer supported, as too much of MC-TESTER depended on ROOT.

Another crucial problem was that expanding functionality of MC-TESTER was limited due to CINT capabilities at that time. CINT was introducing random errors in scripts, to the point when comments influenced code execution. This made it near-impossible to debug such scripts.

Later, when ROOT and CINT became more stable, MC-TESTER could be extended with new functionality. A module for writing user histograms had been added. Users could add them in the form of C++ scripts and operated on the abstract layer of MC-TESTER to prepare data for the histogram.

MC-TESTER is an example of a project that sprouted in response to the need of a validation tool for Monte Carlo generators, and, despite its potential, never really outgrew its initial goals. Over the years, its functionality has greatly expanded and its interface has been enhanced to allow more versatile use. While the aim was to provide the tool to the larger audience, this goal was never reached.

We have noted in the “Physics Background” section how changes introduced to an event record can impact the development of the project. The non-tree-like decay structures could break an algorithm that “did not know” that these entries should not be processed. These algorithms were entering loops as they were expecting a tree-like structure. Results produced by a tool may be invalid for other reasons, such as duplicated entries, where a second entry was treated as a “history” entry. Such defects were trivial to fix. Others, such as the introduction of multiple mothers (more than two) were not as easy. With time, our project became more and
more robust to such external factors. This required constant monitoring of the quality and variation of the input data.

MC-TESTER shows a contrasting case with its strong dependency on ROOT in the early days of the project. Its lack of robustness to the changes implemented in ROOT led to significant problems in its development.

**EVALUATION OF CODE AND TESTS**

Let us provide some information on the evolution of our projects. Depending on the project, the following estimates of complexity are used.

- Code size—number of lines of the project’s source code.
- Event record interface size—number of lines of the project’s interface to event records and algorithms used to traverse event record structure.
- Test code size—number of lines of the executable programs used to test the project and validate the project’s results.
- Results files size—total file size of the test results stored in form of histograms, plots, websites, or papers.

While none of the above metrics can be used to correctly represent the complexity of the scientific software, we have not found a metric that can do so adequately. We feel, however, that together these quantities indicate the overall effort needed to produce a new software version with increased precision of the results.

Figure 2(b) in the supplementary material, available online, summarizes how the code-base of the projects have evolved. This information is provided in detail in Table 1, which presents project evolution as a function of the precision tag. All estimates provided in these tables are indicative. Sources of these values, as well as references to the related publications, are listed in Appendix D and this information is summarized graphically in Figure 3 in the supplementary material, available online.

**SUMMARY**

In this article, we have analyzed long-term phenomenological computing projects from the perspective of software evolution. We have not referred to progress in physics, which was the central point of these projects, beyond the essential minimum. Our aim was to address the nonphysics community, in particular, people who can be involved in similar projects from different domains. These projects, despite large programming efforts, are mostly focused on the domain in which the research is performed. As such, minimum effort is given to the development strategies and evolution of the software of these projects.

We attempted to define milestones in the development of these projects through reaching new precision thresholds. We attempted to quantitatively show that with higher precision demand, more effort is put into writing and maintaining the testing and validation environment than into development of the project itself. We also outlined the most common tests performed for physics software as well as presented two testing techniques that proved useful during the development of these projects.

We have often stressed how much effort goes into building and maintaining the testing environment. The demand for testing and validation of the results increases with higher precision demand. In some cases, the tests for one project evolve as by-products of previous projects.

We also observed how some of our projects were impacted negatively as a consequence of our action or inaction. That is why we described the most important lessons we learned from all of our projects, including the less successful ones. Errors made during the development process are painful and no one wants to remember them. However, it is always important to keep them in mind as they may help in the development of future projects. We have discussed here several examples of such failures, and all of them helped us to avoid similar results in subsequent projects.

**Looking Ahead**

It is always important to look ahead for future extensions and applications. At each step of the evolution of the project, one should always consider what parts of the project might be needed as a starting point for the new work. As the projects...
described in this article show, new projects often emerge as a by-product of the work on another. The development of the software meant to be used for scientific analyses can span over decades. Teaching and motivating new participants is essential for the mature projects to survive generational change.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Centre, Poland, under Grant UMO-2017/27/B/ST2/01391 and in part by the AGH Grant.

REFERENCES


**Tomasz Przedzinski** is currently an associate professor with Jagiellonian University, Krakow, Poland. His research interests include software engineering, software architecture, and applied computing science. Contact him at tprzedzi@cern.ch.

**Maciej Malawski** is currently an associate professor with AGH, Krakow, Poland. His research interests include parallel and distributed computing, cloud infrastructures, and scientific workflows. Contact him at malawski@agh.edu.pl.

**Zbigniew Wąs** is currently a professor with IFJ PAN, Krakow, Poland. His research interests include phenomenology of accelerator particle physics and Monte Carlo methods. Contact him at wasm@mail.cern.ch.
Staged Computation: The Technique You Did Not Know You Were Using

Konrad Hinsen
Centre de Biophysique Moléculaire in Orléans,
Synchrotron Soleil

A quick web search for “staged computation” will convince you that it refers to an exotic technique of interest mainly to programming language designers and implementers. Nothing could be further from the truth. It is a technique that everybody is using all the time, and it is one of the main reasons why reproducible results are so difficult to achieve. In other words, it is something that every computational scientist should know about.

By definition, a staged computation is a computation that proceeds as a sequence of multiple stages, in which each stage produces the code for the next stage, except for the last stage that produces the final result. Most of the references you will find in a Web search are about the so-called metaprogramming techniques, in which a program’s source code is transformed before actually being compiled into an executable. The most widely used metaprogramming technique in scientific computing is the use of macros in the C and C++ languages. These macros are rules for rewriting the source code before compilation that are used for specializing the code, or for adapting it different platforms. Other languages have more elaborate metaprogramming tools, in particular the languages of the Lisp family.

However, the kind of staged computation that I will discuss here is something quite different: It is the very use of compilers. A compiler transforms a program from one notation (“source code”) to another notation (“executable binary”). Executing a piece of source code thus requires two stages: the first stage, compilation, produces the code that is run in the second stage, execution. But that is not the end of the story. The compiler you run is typically an executable binary stored somewhere in your computer’s file system, and so are the libraries that a program makes use of. These binaries have been produced by someone else, on another computer, using yet another compiler. So our two-stage computation is really
a many-stage computation, with the results of the initial stages stored on some server for downloading by people like you. Package managers, such as apt used by Linux distributions such as Debian or Ubuntu, or Homebrew for macOS, make this approach straightforward in practice.

MULTISTAGE REPRODUCIBILITY

Figure 1 provides a visual illustration of a staged computation. Each box in this diagram corresponds to data stored in a file, but there are three distinct categories of data. The first category, shown in blue, is input authored by humans, i.e., mostly program source code. The second category, shown in magenta, is observational input, typically coming from the experimental equipment. The third and dominant category, shown in green, is result of computations. The computations that produce them are indicated by arrows that link inputs to outputs.

What we would like to be reproducible is the box in the bottom right, i.e., the figures and tables we put into our publications. The important message of Figure 1 is that reproducing these results requires all the other items in the diagram to be precisely identified and either archived or themselves reproducible. And since even a modest computation can accumulate hundreds of little boxes, this is not a trivial requirement.

Let us look at what this means for our three categories of items. Human input is the easiest case: It cannot be reproduced, so it must be archived. Note that “archiving” means more than just storing a copy in a safe place. That would be a backup, not an archival copy. Archiving requires producing a safely guarded copy plus a handle via which this copy can be retrieved unambiguously. That handle could be a file name or a URL (both very fragile), a Digital Object Identifier (DOI), which is already more robust, or ideally a handle computed from the content itself,1 such as the identifiers used by Software Heritage.2 Since human input tends to evolve in the course of a research project, it is also advisable to keep it under version control.3 Both Zenodo (https://zenodo.org/) and Software Heritage (https://www.softwareheritage.org/) provide facilities for easily archiving version-controlled human input.

It is more difficult to give general recommendations for observational input, because that is a very diverse category. Like human-authored input, observational input is not reproducible, and must therefore be archived. For choosing an adequate archiving technique and platform, the nature and size of the data matters, but also legal criteria such as ownership or privacy.

For the computed results, in particular compiled software, we have the choice between archiving and reproducing. Unfortunately, the tools we have been using for decades to manage software support neither option satisfactorily. They have been designed for installing, updating, and deploying software, but not for tracking provenance or reproducing earlier states. These tasks must therefore be assumed by humans, who are not very good at managing hundreds of items. This is why reproducible computations remain such a tough challenge. The good news is that computers are very good at dealing with large problems, meaning that we can delegate the management of staged computations to software tools, as I will show later.

STATE OF THE ART

Consider the very simple program shown in Figure 2, which does a few computations and prints the results. On a typical Linux system, you would run it using
gcc pi.c -o pi
./pi
assuming you are using the popular GNU Compiler Collection. The few lines of text printed by
the program are the final result. That is what goes into the bottom-right box in Figure 1. The executable binary pi is the “Program” in the box on the left end of the arrow. There is no “Input data” in this case. The source code file pi.c is the “Source code” in the box above. The “Compiler” is gcc and it looks like we have no libraries, so we have properly identified everything in the rightmost three boxes of Figure 1. That is a good start!

The bad news is that appearances are deceptive: We do have libraries, the compiler is merely hiding them from us. Under the hood, the compiler runs additional programs such as as or collect2, and adds libraries from the C language runtime system. We need to add this hidden stuff, with version numbers, to the “Compiler” and “Library” fields of the box. And with compilers playing hide-and-seek, we need a more reliable way to figure out all of our dependencies!

I suspect that many readers think that I am exaggerating. We are talking about a minuscule C program. All it takes to run is a toolchain for compiling C programs. It does not matter if I use version 7 or version 9 of gcc. It matters even less what the version number of collect2 is, whatever it may do. All the stages before the compilation of pi.c should not matter at all. If the gcc that I am running has passed its test suite, all should be fine.

This reasoning is, in fact, correct most of the time, when applied to C compilers and other stable tools and libraries. But most of the time is not all of the time. One particular subtle point is floating-point arithmetic, which has the reputation of being fundamentally irreproducible. And yet, at the level of the operations defined by the standard IEEE-754 (which all processors and compilers today respect), floating-point arithmetic is perfectly deterministic. The problem is that programmers do not write their code in terms of IEEE-754 operations. The C language does not even give access to that level. It is the compiler that generates those low-level instructions, and it assumes that the programmer does not care about the differences due to round-off. Therefore, if you want reproducible floating-point results, their full specification is your code plus the C compiler plus the compilation options.

More importantly, the “details do not matter” reasoning fails for large software assemblies, in which pieces you have never heard about but which have an impact on the final results may change because of bugs or voluntary decisions to break backward compatibility. Under the Guix package manager, about which I will say more below, running the C program in Figure 2 requires four packages (gcc, binutils, glibc, ld-wrapper, the last one being a Guix-specific package). That is what takes the green-colored slots in the rightmost “Compiler” box in Figure 1. For a Python script using the popular NumPy and SciPy extensions, that is already 24 packages. If you include all the packages from the earlier stages of the computation, you get 89 packages for a C program, but 501 for Python + NumPy + SciPy. Figuring out which of these packages should be irrelevant details becomes a serious challenge.

Let us move on for now, assuming that we can somehow keep track of hundreds of dependencies. We must then either archive the compiled code, or make it reproducible, but we can immediately eliminate “archiving” because there is no practically usable infrastructure for this. Sure, package managers download compiled code from servers, but these servers are caches, designed for increasing the efficiency of software distribution. They are not archives from which binaries could be retrieved at arbitrary later times via an unambiguous handle. That also applies to services such as DockerHub that hold container images.
The situation looks more promising for making all compiled code reproducible. Package managers are based on build recipes, which is the code that is run to rebuild the package. Likewise, containers are usually built from such recipes, e.g., the well-known Dockerfiles. Unfortunately, a closer look reveals that most of these build recipes are not reproducible. They say something like “download the current Python source code and compile it with the current version of gcc.” That is great for regularly updating software, which is after all what package and container managers were designed for, but it is not reproducible.

GUIX TO THE RESCUE

The Guix package manager (http://guix.gnu.org/) for GNU/Linux was designed from the start with reproducibility in mind, and offers the best support for reproducible research that is currently available. More specifically, it is based on two fundamental concepts:

- An explicit representation of the full staged computation graph of Figure 1, for all software packages, containing all information that can potentially impact results, and referring to specific versions of all source code.
- Execution in restricted environments. Guix can run programs in environments where only explicitly listed software is available, providing a guarantee that the programs have no other dependencies. These restricted environments are used for building software packages, but are also available to users for running their own code.

To run the example from Figure 2 in a restricted environment in Guix, I would type

```bash
$ guix environment
    -- --ad-hoc gcc-toolchain

[env] $ gcc pi.c -o pi
[env] $ ./pi
```

The first line creates a restricted environment with access to a single software package (gcc-toolchain), a single directory (the current one), and no network access at all. It then starts a shell in that environment, into which the following two lines are typed. The option—container provides the strongest possible isolation of that environment, but less restrictive versions are also available. The fact that my program works correctly in that environment proves that it has no dependencies other than gcc-toolchain, which is a package specifically designed for C programming and contains the gcc compiler plus the utilities and libraries it requires to function. To make computations in such environments reproducible, all I need to do is note the exact version of Guix that I am using

```bash
$ guix describe --f recutils
name : guix
url : https :// git.savannah.gnu.org/git/guix.git
commit : 769b96b62e8c09b078f73adcc09f6b860505920f8f
```

The Guix version is given by the commit field, which is a unique and persistent handle. I can then at any time in the future reproduce the environment, and thus my computation:

```bash
$ guix time-machine
    -- --commit=769b96b62e8c09b078f73adcc09f6b860505920f8f

environment -- --ad-hoc gcc-toolchain
[env] $ gcc pi.c -o pi
[env] $ ./pi
```

If you think that the first line is rather long, consider that it replaces a whole Docker container!

Like other package managers, Guix downloads precompiled binary versions of its packages from a caching server if available. Otherwise, or upon explicit user request, Guix recompiles everything from source code. Well, almost everything: A minimal archived core package called the bootstrap seed is always downloaded in binary form. It contains a basic compiler that is used to get the staged computations started. You cannot avoid having to download that bootstrap seed in binary form, but if you are particularly paranoid, you can then recompile it yourself, and verify that you get the same files, bit for bit.

Since Guix stores the complete staged computation graph, you can also explore it using Guix’ command line tools for standard tasks such as querying the version numbers of the packages. For more advanced needs, you can write scripts in Guile, the dialect of Scheme that Guix is written in. This is what I did to obtain the dependence counts that I quoted earlier. For readers interested in the technical details, there is a post on the Guix blog. 6
At this time, Guix is still a tool for early adopters. Its package collection is not nearly as complete as those of well-established distributions, and its tools still evolve rather rapidly. However, it shows that reproducible staged computations are possible, and it is actually already very usable in practice if all the software you need is in there. Check it out for yourself!

REFERENCES


Konrad Hinsen is a researcher with the Centre de Biophysique Moléculaire in Orléans and at the Synchrotron SOLEIL in Saint Aubin. His research interests include protein structure and dynamics and scientific computing. He received the Ph.D. degree in theoretical physics from RWTH Aachen University, Aachen, Germany. Contact him at konrad.hinsen@cnrs.fr.
Traditionally, hardware description languages (HDLs), such as Verilog or VHDL, have been used for programming Field-Programmable Gate Arrays (FPGAs). However, this approach requires an advanced knowledge of digital design and computer architecture. Recently emerged high-level design tools make it easier for the programmers to code complex designs in C/C++. High-level synthesis (HLS) and OpenCL are the two leading high-level design platforms that are becoming widely used for programming FPGAs. Their proponents claim that these tools require little to no knowledge of the hardware design principles and can significantly improve developer’s productivity. In this article, we explore these two high-level design approaches from the point of view of a software developer. We use Xilinx Vivado HLS C/C++ ver. 2019.1 and Xilinx SDAccel OpenCL ver. 2019.1 to implement a cross-correlation operation from scratch and synthesize it for a Xilinx u250 Alveo FPGA board. The selected operation is at the core of convolutional neural networks and is generally nontrivial to implement using a traditional HDL methodology, but is rather simple to implement using a programming language, such as C. We opted not to focus on the design optimization, but rather getting a design that works on the FPGA with the minimal time spent on its implementation. We use the Xilinx SDAccel platform that provides support for implementing both OpenCL- and HLS-based kernels to run on an FPGA using OpenCL drivers on the host platform. We also took note of how capable each tool is in terms of optimization,
without using tool-specific attributes or pragmas, as well as how well it utilizes the available hardware. We find that HLS tools are easy to learn and the time to design is much shorter compared to the HDL approach. However, a good knowledge of digital design and the underlying FPGA architecture is still needed to deliver a high-performance implementation.

OVERVIEW OF THE TOOLS

The OpenCL kernel was implemented using the Xilinx SDAccel environment and the C-based HLS kernel was implemented in Xilinx Vivado HLS, and then brought into the SDAccel environment to integrate with the host side of the application.

Xilinx Vivado HLS

The Vivado HLS compiler is used to convert code written in C/C++ or SystemC into a register transfer level (RTL) representation, which can then be synthesized to run on an FPGA. From the point of view of the end-user, the HLS compiler is similar to other language compilers that are available for application development. This tool is simple to learn and use, as long as the designer has a good knowledge of C or C++. The overall design approach consists of the following steps.

1. Compile, execute, and debug the algorithm written in C/C++.
2. Synthesize the C/C++ algorithm into an RTL implementation, optionally using user-guided optimization directives.
3. Generate comprehensive reports and analyze the design.
4. Optimize the design to meet application requirements.
5. Verify the Register Transfer Language (RTL) implementation using a pushbutton flow.
6. Package the RTL implementation into a selection of Intellectual Property (IP) formats.

The last step is to package the RTL output files as an IP core. There are three formats they can be packed into depending upon the way in which the IP is going to be used further in the design process. In our case, we packed IP as a Vivado IP catalog format and exported the generated .xo file into SDAccel. This .xo file, which contains the actual kernel, will be integrated with the OpenCL FPGA wrapper by the SDAccel tools and can be loaded onto the FPGA from the code executed on the host.

Xilinx SDAccel

The OpenCL-based design flow utilizes SDAccel, an Eclipse-based integrated development environment. We use C/C++ with OpenCL API calls for the host software implementation, and OpenCL for hardware kernel development. The host application is built through using the standard gcc host compiler, and the FPGA binary is built through a separate process that uses the Xilinx xocc compiler. The overall design approach consists of the following steps.

1. Code the desired kernel in OpenCL.
2. Run software emulation to ensure functional correctness.
3. Run hardware emulation to generate host and kernel profiling data.
4. Optimize kernel for performance using directives and code restructuring techniques.
5. Run hardware synthesis to generate the FPGA kernel bit file.
6. Write host code using the OpenCL API to interface with the kernel executed on the FPGA.

Note that nearly the same host code can be used to interface with the FPGA kernel regardless of the way the kernel was produced, as long as it was imported into SDAccel. Therefore, we use the same host code to test both our designs.

OpenCL IMPLEMENTATION

The source code of the cross-correlation kernel implemented in OpenCL is shown in Figure 1. The host code writes the data required by the kernel into a memory bank on the FPGA board. The kernel code then performs the computation by accessing the data from that memory. Once the kernel execution completes and the data are written back to the FPGA-attached memory, the host code reads the data back from the global memory and continues processing as needed.

The input image, filter mask, and output image are declared as global variables in our
kernel. Along with these variables, the image and mask dimensions are also declared as function arguments. The values for the function arguments are written into the global memory by the host code and are read in during kernel execution. The design consists of four nested loops that are responsible for moving the mask across the image horizontally and vertically. The design uses a stride and padding equal to 1.

Compared to a straight OpenCL code, our kernel uses one additional line of code, __attribute__((reqd_work_group_size(1, 1, 1))), which specifies working size of the kernel to be just one copy.

Once the host code and kernel code are created, the next step is to build the application. The build target can be chosen by the programmer. The SDAccel provides three such targets: software emulation, hardware emulation, and system. Our design was compiled and built using the system option targeting the Xilinx u250 Alveo FPGA board. The SDAccel generates reports that can be used to analyze the timing, latency, and area information of a design.

From the timing information we can, for example, verify that the estimated frequency is same as the target frequency. The area information can be used to further guide the optimization of the design.

**VIVADO HLS C/C++ IMPLEMENTATION**

In the HLS implementation, the hardware kernel is created in C/C++ using the Vivado HLS tool. The optimization and performance validation are done in HLS. The major difference between OpenCL and Vivado HLS C/C++ kernel implementation is the use of HLS pragmas instead of OpenCL attributes. These pragmas are used to declare appropriate interfaces for scalar and vector variables (see Figure 2). The kernel acts as an accelerator in SDAccel and is required to be modeled using the guidelines provided by SDAccel. Interfaces are modeled as Advanced eXtensible Interface (AXI) memory interfaces and scalar parameters called by the value are mapped to an AXI4-Lite interface. When creating interfaces, it is important to specify the depth of the AXI ports—the wrong depth size will result in a C/RTL simulation mismatch.

Once the kernel design is complete, the C simulation tool can be used to verify the design. A simple testbench can be written to run the C simulation. Once the functionality of the design is verified, the C Synthesis tool is used to synthesize the design to an RTL implementation. A top-level function is required in the tool. The

![Figure 1. OpenCL kernel implementation.](image-url)
C Synthesis tools have several options to make this simpler, such as echoing the progress of the synthesis project to console and a GUI interface that provides enhanced information hyperlinks in the output messages, which provide more information on the source of design issues and how to resolve them. When synthesis is complete, a report for the top-level function is generated. The report provides details on both the performance and the area of the RTL design and can be used to guide further performance optimizations.

After successfully synthesizing the design, the next step is to verify if the RTL is correct. This is done using the C/RTL co-simulation tool. When the verification completes, the last step in the Vivado HLS design flow is to package the RTL output as an IP. This is done using the Export RTL tool that packages the RTL as a Xilinx object (.xo) file that can then be included in the SDAccel project. Thus, when creating a new project in the Vivado HLS, it is important to select the SDAccel Bottom Up Flow option, which allows one to run HLS kernels in SDAccel. A host code similar to the one used in the OpenCL implementation can be used to run the HLS kernel on the FPGA.

DISCUSSION

We have shown how to implement the cross-correlation operation with the two programming frameworks. The biggest challenge was to learn how to use the tools and how to implement a host code using API calls for creating platforms, attributes, and contexts for executing a kernel on the FPGA hardware. The design tools are rather complex and the code implementation

```c
extern "C" {
void convolution_hls(int *input_img, // Read-only input Image
    int *input_krn1, // Read-only input Kernel
    int *output_img, // Output Image
    int dim_img, // Input Dimension
    int dim_krn1 // Kernel Dimensions
) {
    #pragma HLS INTERFACE m_axi port=input_img offset=slave depth=9 bundle=gmem
    #pragma HLS INTERFACE m_axi port=input_krn1 offset=slave depth=4 bundle=gmem
    #pragma HLS INTERFACE m_axi port=output_img offset=slave depth=4 bundle=gmem
    #pragma HLS INTERFACE s_axilite port=input_img bundle=control
    #pragma HLS INTERFACE s_axilite port=input_krn1 bundle=control
    #pragma HLS INTERFACE s_axilite port=output_img bundle=control
    #pragma HLS INTERFACE s_axilite port=dim_img bundle=control
    #pragma HLS INTERFACE s_axilite port=dim_krn1 bundle=control
    #pragma HLS INTERFACE s_axilite port=return bundle=control

    int sum;
    ap_int<8> dim_out=dim_img-dim_krn1+1;

    loop_1: for(int y=0; y<dim_out; y++) {
        loop_2: for(int x=0; x<dim_out; x++) {
            sum=0;
            loop_3: for(int j=0; j<dim_krn1; j++) {
                loop_4: for(int k=0; k<dim_krn1; k++) {
                    sum += input_img[(y+j)*dim_img+x+k]*input_krn1[j*dim_krn1+k];
                }
            }
        output_img[y*dim_out+x]=sum;
        }
    }
}
```

Figure 2. HLS C kernel implementation.

July/August 2020
methodology is tailored more toward hardware designers that software programmers.

The kernel code very closely resembles the original C code, with some additional directives for guiding the compiler to properly implement the intended design. However, the kernels we have implemented were not optimized, and only basic attributes or pragmas were applied to enable proper functionality. Our OpenCL implementation consists of 29 lines of kernel code and 131 lines of host code, whereas the HLS kernel consists of 40 lines of kernel code and 154 lines of host code. (Our host code also included a CPU version of the kernel for verification.) Compared with a traditional CPU-only implementation, this is a similar code size.

A typical compilation time to generate the complete FPGA design even for such a small kernel is well over three hours on a multicore system where multiple cores are used by the Xilinx tools. This is, of course, a significantly longer compilation time than software designers are used to in more common environments.

Even though we have not applied any manual optimizations, the compilers attempt to flatten, unroll, and pipeline loops on their own, albeit with limited success. For example, as reported by HLS synthesis tools, loops 3 and 4 are flattened and pipelined with a depth of 21 cycles and a loop initiation interval of 2. The limiting factor for the loop initiation interval is the fact that the data is coming from a single memory port that can only provide one input value on each clock cycle. Both designs are reported to work at 300 Mhz.

The kernel could be further optimized using attributes or pragmas and code restructuring. The OpenCL approach is somewhat easier for software programmers who are not familiar with hardware design, but even this approach is not entirely friendly for those not familiar with the hardware design terminology. The application developer needs to be able to read hardware synthesis reports and correlate the information from these reports with their design. The tools produce a substantial amount of reports in each phase of the code compilation process, and each of these reports contains information instrumental in guiding the kernel optimization. These reports are oriented toward hardware designers.

Can software developers without knowledge of hardware design principles use these platforms to develop codes for FPGAs? Yes, but they will have to learn a lot in the process. The Vivado HLS approach is quite involved with all the verification steps, IP generation, integration with SDAccel project, etc., whereas OpenCL is easier for software developers. However, even a naı̈ve implementation, which is what we have done here, is relatively involved and time-consuming regardless of the framework used. And of course, any performance optimizations will require an understanding of the specific FPGA board’s architecture. For example, efficient utilization of the available memory bandwidth on the u250 Alveo FPGA board requires distributing data across four DDR memory banks as well as defining global pointers as 512-bit data types to sustain the maximum data bandwidth between the kernel and DDR memory. This requires changes to the host and kernel code to specify which bank is to be used for which data buffer as well as splitting data into several buffers. Software developers are almost never concerned with such issues.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation’s Major Research Instrumentation program under Grant 1725729, as well as the University of Illinois at Urbana-Champaign.

REFERENCES

Ramshankar Venkatakrishnan is a Research Programmer at the National Center for Supercomputing Applications. He received the master's degree from University of Illinois at Chicago in 2015. Contact him at rvenka21@illinois.edu.

Ashish Misra is a Postdoctoral Research Associate at the National Center for Supercomputing Applications. He received the master’s and Ph.D. degrees from Birla Institute of Technology and Science, India, in 2008 and 2017, respectively. Contact him at ashishm@illinois.edu.

Volodymyr Kindratenko is a Senior Research Scientist at the National Center for Supercomputing Applications. He received the Specialist degree from the Vynnychenko State Pedagogical University, Kirovograd, Ukraine, in 1993 and the D.Sc. degree from the University of Antwerp, Belgium, in 1997. Contact him at kindrtnk@illinois.edu.
The HPC Certification Forum: Toward a Globally Acknowledged HPC Certification

Julian Kunkel  
University of Reading

Weronika Filinger  
University of Edinburgh

Christian Meesters  
Johannes Gutenberg-Universität Mainz

Anja Gerbes  
Goethe-Universität Frankfurt am Main

Abstract—The goal of the HPC Certification Forum is to categorize, define, and examine competencies expected from proficient HPC practitioners. The community-led forum is working toward establishing a globally acknowledged HPC certification process, a process that engages with HPC centers to identify gaps in users’ knowledge, and with users to identify the skills required to perform their tasks. In this article, we introduce the forum and summarize the progress made over the last two years. The release of the first officially supported certificate is planned for the second half of 2020.

The ever-changing nature of high-performance computing (HPC) has always compelled the HPC community to invest in continual efforts to train new and existing practitioners. Historically, these efforts were tailored around a typical group of users possessing, due to their background, a certain set of programming skills. However, as HPC has become more diverse in terms of hardware, software, and user backgrounds, the traditional training approaches have become insufficient to address the training needs of our community. An increasingly complicated HPC landscape makes the development and delivery of new training materials challenging. During training delivery, educators need to address the knowledge gaps resulting from the diverse backgrounds of their learners. It is not uncommon for an attendee to come with a
specific learning objective related to their work tasks and not be interested in the core HPC knowledge. In that sense, the term HPC practitioner describes anyone involved in providing or using HPC systems, e.g., a user that runs an application on an HPC-resource, a developer for HPC-systems, or an administrator. At the same time, we define the term “HPC” inclusively, capturing parallel computing and cluster computing, e.g., high-throughput computing or multitask computing, as these too suffer from a lack of knowledge with regard to performance issues.

How should we develop training for users, often coming from disciplines that have not traditionally used HPC resources, and are only interested in learning a particular set of skills? How can we satisfy their training needs if we do not really understand what these are? HPC centers struggle to identify and overcome the gaps in users’ knowledge, whereas users struggle to identify the skills required to perform their tasks.

The goal of the HPC Certification Forum (HPCCF) is to clearly categorize, define, and examine competencies expected from proficient HPC practitioners. The HPCCF is the central authority, and curates and maintains the certification program. The program consists of three parts: competencies defined in a modular and easily expandable skill tree, an examination process to verify that practitioners possess those skills, and the certification demonstrating their knowledge. Although the forum is not involved in the development of any training materials or tools, it supports the ecosystem around the competencies.

The ultimate goal of the forum is to offer a free, globally acknowledged certification program that will make HPC education and training more transparent and quantifiable for training providers, and easier to navigate for practitioners. This article highlights relevant aspects of our activities, more details can be found in.¹

COMMUNITY-LED FORUM

The forum is organized around several key roles, which include: the general chair, a publicity chair, and curators for the skill-tree, topics, and examinations. While the board leads the effort, members of the community are expected to contribute to the effort, and anyone is free to benefit from it. Active members can gain nomination and voting rights via an annual steering board election. Decision making is lightweight at the moment: while we have defined roles for steering board members that include final authority in the event it is needed, thus far we have made decisions democratically without the need to rely on this formal mechanism. Basically, any contribution is either accepted or discussed and modified until it is accepted.

The forum uses Slack for its monthly meetings and organizes two face-to-face meetings per year (one at ISC-HPC and the other at the annual SC). GitHub and the Forum’s webpage are used to coordinate the effort and publish information.

All software used by the forum is Open Source and freely available to allow everyone to participate. The forum aims to provide an ecosystem revolving around the certification specification (including the skill tree and the examination framework), which consists of tools that cover, for example, branding of teaching materials, referring and cross-linking to the competency definitions, and compiling curricula. In particular, we hope to catalogue and reference the existing content of third-parties to allow practitioners to browse the skills and navigate to relevant open and commercial teaching material.

Note that there is currently no direct funding for the effort, but we support all proposals and efforts that members bring forward and associate their work with the forum. For example, in the ESiWACE project, some contributions regarding HPC IO are expected. Ultimately, we believe that the sustainability of the effort depends upon the recognition of its importance and the voluntary contribution of institutions and individuals.

CATEGORIZATION OF COMPETENCIES

The forum groups a well-defined set of competencies into a skill, and a skill is identified by a set of learning outcomes and relevant metadata that clearly specifies what a practitioner should be able to do to be said to possess that skill. The skills are organized in a tree structure from a coarse-grained representation (corresponding to the tree branches) to a fine-grained representation mapped onto the tree leaves. On the leaf
level, a skill is orthogonal to other skills—their narrowed scope means they intentionally can be taught in sessions ranging from a 1.5-hour lecture up to a 4-hour workshop. Skills may cover technology-specific knowledge, such as the skill “USE1.1-B Command Line Interface” for Linux basics or the skill “K4.2-B SLURM Workload manager,” which describes how a cluster manages user jobs. We believe this granularity allows practitioners to select skills relevant to their circumstances, and allows lecturers to prepare modular training sessions with well-defined content while still achieving comparable training outcomes for a varied range of practitioners’ backgrounds. Cross-linking between skills belonging to different branches is allowed and provides for the reuse of the skill definitions and eases the navigation of the tree.

EXAMINATION

As the Forum aims to keep the certification free for practitioners, an online examination process has been chosen. The summative assessment will be conducted primarily using multiple-choice questions. For each skill a pool of questions and answers will be created, drawing from both internal and external contributions, and an examination will consist of randomly selected questions. Future developments will include questions beyond the multiple-choice type.

We believe the incentive to deliberately cheat during the assessment, e.g., by having the exam filled by someone else, is low. Therefore, we address this issue in a lightweight and cost-effective fashion. Our process deploys several strategies to minimize the risk of cheating, such as raising the examinees’ awareness, using a large pool of questions, setting time limits for each question, and a delay between registering for and taking the actual examination. Since knowledge can quickly become obsolete, each certificate needs to indicate when (month and year) the qualifying examination took place. Also, because the examination of a single fine-grained leaf-level skill would be too easy to pass with short-term memorization and more prone to cheating, the certificates bundle multiple skills together. To ultimately provide trust, the Forum hopes to provide the automatic generation of short tests for prospective employers that would allow validating the knowledge of applicants under their own supervision or in assessment centers.

We manage exam questions internally within the HPCCF consortium. To facilitate external contributions, we provide on our wiki-webpage a light-weight interface for suggesting a question for each skill-set which can be used by anyone—those skill-related questions are reviewed and ultimately managed by the exam curator who is part of the steering board. A well-defined process is being created for prospective contributors to verify the suitability of the contributed questions. For a question to be approved, it will need to undergo a review process where comprehensibility, logic (does it have at least one clear answer?) and rigor (does the question lead to the expected answer?) have to be met. Particularly, each exam will be designed considering a variety of aspects, such as competency assessment, learning goals, exam objectivity, and taxonomy of educational objectives.

As HPCCF exams are potentially interspersed between courses for a particular examinee, examinees may profit from the forward learning effect, i.e., the outcome of such a lightweight test may influence how a practitioner will study the material further. Lecturers, who will be provided with “their” examinees summarized outcomes, may profit from insights to conceptual issues course participants may exhibit.

The questions are the only proprietary component of the HPCCF—using restrictive license terms for authors while giving them credit. By providing potential employers with our tests, too, will have the means to test how an exam is seen and complete it from a student’s point of view; this is a measure with established relevance.

RELATED EFFORTS

The certification program is a new community-wide effort and to the best of our knowledge nothing similar was attempted at such a scale, both with respect to the comprehensibility of the covered content and the international reach of the effort. The other related efforts mainly focus on either providing a comprehensive catalogue of existing training materials and opportunities, providing simple badges confirming participation in a
specific training event, or establishing a branded well-defined content and teaching practices recognized by the community. A number of institutions and organizations attempted to catalogue the existing training materials, keep a list of training events and bring the HPC training community together, the ongoing efforts include: PRACE Training Portal, the HPC University and the ACM SIGHPC Education Chapter. An example of a badging effort is XSEDE Training Badges Program. Finally, the last example refers both to the Carpentries initiative, and HPC Carpentry which was developed in recent years. The HPC Certification Forum recognizes the importance of these efforts and is actively engaging with their contributors.

CONCLUSION

The program of the HPCCF allows the existing content to be reused but also makes it possible to create a new ecosystem in which HPC centers, research labs, academic institutions, and commercial companies could offer the best of their teaching material. The HPCCF aims to support existing activities and complements them by providing a unified and clear way of mapping out the relevant HPC competencies. It should be emphasized that the HPCCF does not regulate the content of training material; we purposely separate the definition of skills, the examination, and the certification from the content delivery. The program does not prescribe a curriculum or any fixed order in which skills should be obtained, thus providing flexibility. It eases the navigation between different competencies without being overly restrictive. We are hoping that a majority of the existing and newly created teaching resources can be branded indicating the skills they cover.

We believe the program will bring multiple benefits to everyone involved in HPC teaching and training. Making clear what skills are required or recommended for a competent HPC user would be helpful to both the HPC service providers and practitioners. Training providers could bundle together skills that are most beneficial for specific user roles and scientific domains, which would allow practitioners to browse through skills to quickly identify and learn the skills required to perform their tasks. The variety of training offered within the HPC community makes finding the right resources more complicated than it should be. We hope that the certification program will eventually provide useful information on where the desired skills are taught. The examination confirming that a certain set of competencies has been acquired makes the learning process more complete and meaningful.

By participating in the program, HPC training providers can increase the visibility of their teaching opportunities and share their resources more effectively. The mapping of the skills defined by the program onto the existing training materials should also help to identify any potential gaps and improve the integrity of the offered training. Finally, the certificates recognized by the whole HPC community will simplify the intercomparison of independently offered courses and provide additional incentives for participation. Overall, the flexibility of the program allows for the construction of more personalized and just-in-time pathways to learn about HPC.

To achieve these goals, the forum welcomes contributions from volunteers. For this initiative to truly fulfill its role the involvement of the members of the HPC training community with diverse backgrounds and experiences is required. As the HPCCF community is managed collaboratively according to self-managed evolving rules, it is welcoming and the expected contribution is not demanding, due to the informal nature of the forum.

REFERENCES

**Julian Kunkel** is currently a Lecturer with the Computer Science Department, University of Reading. He was a Postdoc with the Research Department, German Climate Computing Center (DKRZ), which partners with the Scientific Computing Group, Universität Hamburg. He manages several research projects revolving around high-performance computing and particularly high-performance storage and teaches parallel and distributed computing. Besides his main research focus, he is involved in several community-building activities and aims to improve the training situation for researchers. Contact him at julian.kunkel@googlemail.com.

**Weronika Filinger** is an HPC Applications Consultant with EPCC, The University of Edinburgh. She has been deeply involved in the design and development of the first Massive Open Online Course (MOOC) on Supercomputing, and facilitated all runs of the course. She is teaching the Practical Introduction to HPC online course, which is offered as a part of an online master’s program at the University of Edinburgh. For several years, she has been a member of the EPCC outreach team, taking HPC-related activities to public events. She is also the Co-Chair of the Outreach Committee of the ACM SIGHPC Education Chapter and the Publicity Chair of the International HPC Certification Program. She is also involved in running the International HPC Summer School. Over the years, she has worked on a number of collaborative projects such as CRESTA, ADEPT, APES, SARGASSO and DEEP-EST, and provided consultancy for the Software Sustainability Institute. Contact her at w.filinger@epcc.ed.ac.uk.

**Christian Meesters** is a Computational Scientist with the HPC Center, Johannes Gutenberg-University, Mainz. He was a Postdoc with the University Hospital, Bonn, where he developed genetic epidemiology workflows and applications to leverage them to HPC environments. He coordinates local courses and teaching efforts. As the site’s life science contact for HPC, he aims to develop generic life science workflows for HPC clusters. Contact him at meesters@uni-mainz.de.

**Anja Gerbes** is with the Center for Scientific Computing and is a member of Hessian Competence Center for High Performance Computing, Goethe University, Frankfurt/Main. A considerable part of her job role is to develop a range of courses and resources to enable users to work with the cluster. The goal of her Ph.D. is to study the compiler for deficits in terms of performance when translating HPC applications and to understand the limitations of compilers in making the necessary optimizations. These insights can then be incorporated into the compiler for future automatic compiler optimization. Automatic program transformation using source-to-source instrumentation of parallel programs will prepare HPC applications for future performance analysis. Contact her at gerbes@fias.uni-frankfurt.de.
PURPOSE: The IEEE Computer Society is the world’s largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine Computer, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEBSITE: www.computer.org

OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):
- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

Computer: The flagship publication of the IEEE Computer Society, Computer publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals: The society publishes 12 magazines and 18 journals. Refer to membership application or request information as noted above.

Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.

Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.

Technical Committees: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society offers three software developer credentials. For more information, visit www.computer.org/certification.

BOARD OF GOVERNORS MEETING

24 – 25 September 2020 in McLean, Virginia, USA

EXECUTIVE COMMITTEE

President: Leila De Floriani
President-Elect: Forrest Shull
Past President: Cecilia Metra
First VP: Riccardo Mariani; Second VP: Sy-Yen Kuo
Secretary: Dimitrios Serpanos; Treasurer: David Lomet
VP, Membership & Geographic Activities: Yervant Zorian
VP, Professional & Educational Activities: Sy-Yen Kuo
VP, Publications: Fabrizio Lombardi
VP, Standards Activities: Riccardo Mariani
VP, Technical & Conference Activities: William D. Gropp
2019–2020 IEEE Division VIII Director: Elizabeth L. Burd
2020-2021 IEEE Division V Director: Thomas M. Conte
2020 IEEE Division VIII Director-Elect: Christina M. Schober

BOARD OF GOVERNORS

Term Expiring 2021: M. Brian Blake, Fred Douglass, Carlos E. Jimenez-Gomez, Ramalatha Marimuthu, Erik Jan Marinissen, Kunio Uchiyama
Term Expiring 2022: Nils Aschenbruck, Ernesto Cuadros-Vargas, David S. Ebert, William Gropp, Grace Lewis, Stefano Zanero

EXECUTIVE STAFF

Executive Director: Melissa A. Russell
Director, Governance & Associate Executive Director: Anne Marie Kelly
Director, Finance & Accounting: Sunny Hwang
Director, Information Technology & Services: Sumit Kacker
Director, Marketing & Sales: Michelle Tubb
Director, Membership Development: Eric Berkowitz

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928; Phone: +1 202 371 0101; Fax: +1 202 728 9614;
Email: help@computer.org
Los Alamitos: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720;
Phone: +1 714 821 8380; Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS
Phone: +1 800 678 4333; Fax: +1 714 821 4641;
Email: help@computer.org

IEEE BOARD OF DIRECTORS

President: Toshio Fukuda
President-Elect: Susan K. “Kathy” Land
Past President: José M.F. Moura
Secretary: Kathleen A. Kramer
Treasurer: Joseph V. Lillie
Director & President, IEEE-USA: Jim Conrad
Director & President, Standards Association: Robert S. Fish
Director & VP, Educational Activities: Stephen Phillips
Director & VP, Membership & Geographic Activities: Kukjin Chun
Director & VP, Publication Services & Products: Tapan Sarkar
Director & VP, Technical Activities: Kazuhiro Kosuge

revised 1 May 2020
ComputingEdge

Your one-stop resource for industry hot topics, technical overviews, and in-depth articles.

Cutting-edge articles from the IEEE Computer Society's portfolio of 12 magazines.

Unique original content by computing thought leaders, innovators, and experts.

Keeps you up to date on what you need to know across the technology spectrum.

Subscribe for free
www.computer.org/computingedge