

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

12/2019 VOL.62 NO.12

The Rise of Serverless Computing

Hack for Hire

Public Entrepreneurship

Malevolent Machine Learning

Automated Program Repair



Association for
Computing Machinery

acm

ANNOUNCING 21 IPDPS 2020 WORKSHOPS

IPDPS Workshops are the “bookends” to the three-day conference technical program of contributed papers, invited speakers, student programs, and industry participation. They provide the IPDPS community an opportunity to explore special topics and present work that is more preliminary or cutting-edge than the more mature research presented in the main symposium. Each workshop has its own website and submission requirements, and the submission deadline for most workshops is after the main conference author notification dates.

WORKSHOPS MONDAY 18 May 2020

HCW	Heterogeneity in Computing Workshop
RAW	Reconfigurable Architectures Workshop
HiCOMB	High Performance Computational Biology
GrAPL	Graphs, Architectures, Programming, and Learning
EduPar	NSF/TCPP Workshop on Parallel and Distributed Computing Education
HIPS	High-level Parallel Programming Models and Supportive Environments
HPBDC	High-Performance Big Data and Cloud Computing
AsHES	Accelerators and Hybrid Exascale Systems
PDCO	Parallel / Distributed Combinatorics and Optimization
APDCM	Advances in Parallel and Distributed Computational Models

WORKSHOPS FRIDAY 22 MAY 2020

JSSPP	Job Scheduling Strategies for Parallel Processing
CHIUW	Chapel Implementers and Users Workshop
PDSEC	Parallel and Distributed Scientific and Engineering Computing
iWAPT	Automatic Performance Tuning
MPP	Parallel Programming Models - Special Edition Machine Learning Performance and Security
SNACS	Scalable Networks for Advanced Computing Systems
PAISE	Parallel AI and Systems for the Edge
RADR	Resource Arbitration for Dynamic Runtimes
ScaDL	Scalable Deep Learning over Parallel and Distributed Infrastructures
HPS	High-Performance Storage
ParSocial	Parallel and Distributed Processing for Computational Social Systems

GENERAL CO-CHAIRS

Anu Bourgeois (Georgia State University, USA)

Ramachandran Vaidyanathan (Louisiana State University, USA)

PROGRAM CHAIR

Yuanyuan Yang (NSF and Stony Brook University, USA)

WORKSHOPS CHAIR and VICE-CHAIR

Erik Saule (University of North Carolina Charlotte, USA)

Jaroslav Zola (The State University of New York at Buffalo, USA)

STUDENT PARTICIPATION CHAIRS

Edson Borin (University of Campinas, Brazil)

Jay Lofstead (Sandia National Laboratories, USA)

PHD FORUM & STUDENT MENTORING

This event will include traditional poster presentations by PhD students enhanced by a program of mentoring and coaching in scientific writing and presentation skills and a special opportunity for students to hear from and interact with senior researchers attending the conference.

INDUSTRY PARTICIPATION

There are several ways for industry to partner with IPDPS and share the benefits of associating with our international community of top researchers and practitioners in fields related to parallel processing and distributed computing. IPDPS is a “walk-up-and-talk” venue that encourages industry partners to use the conference as a way to introduce their technology in an informal setting.

IMPORTANT DATES

Conference Preliminary Author Notification

- December 9, 2019

Workshops' Call for Papers Deadlines

- Most Fall After December 9, 2019

IPDPS 2020 VENUE

New Orleans is one of the most eccentric and lively cities in the world. Whatever your interests are, New Orleans has you covered. From its diverse culture, distinctive cuisine, rich history, colorful celebrations, live music, vibrant nightlife, and world-class restaurants, there is something for everyone. Join IPDPS in 2020 to find out what makes New Orleans so unique and special.

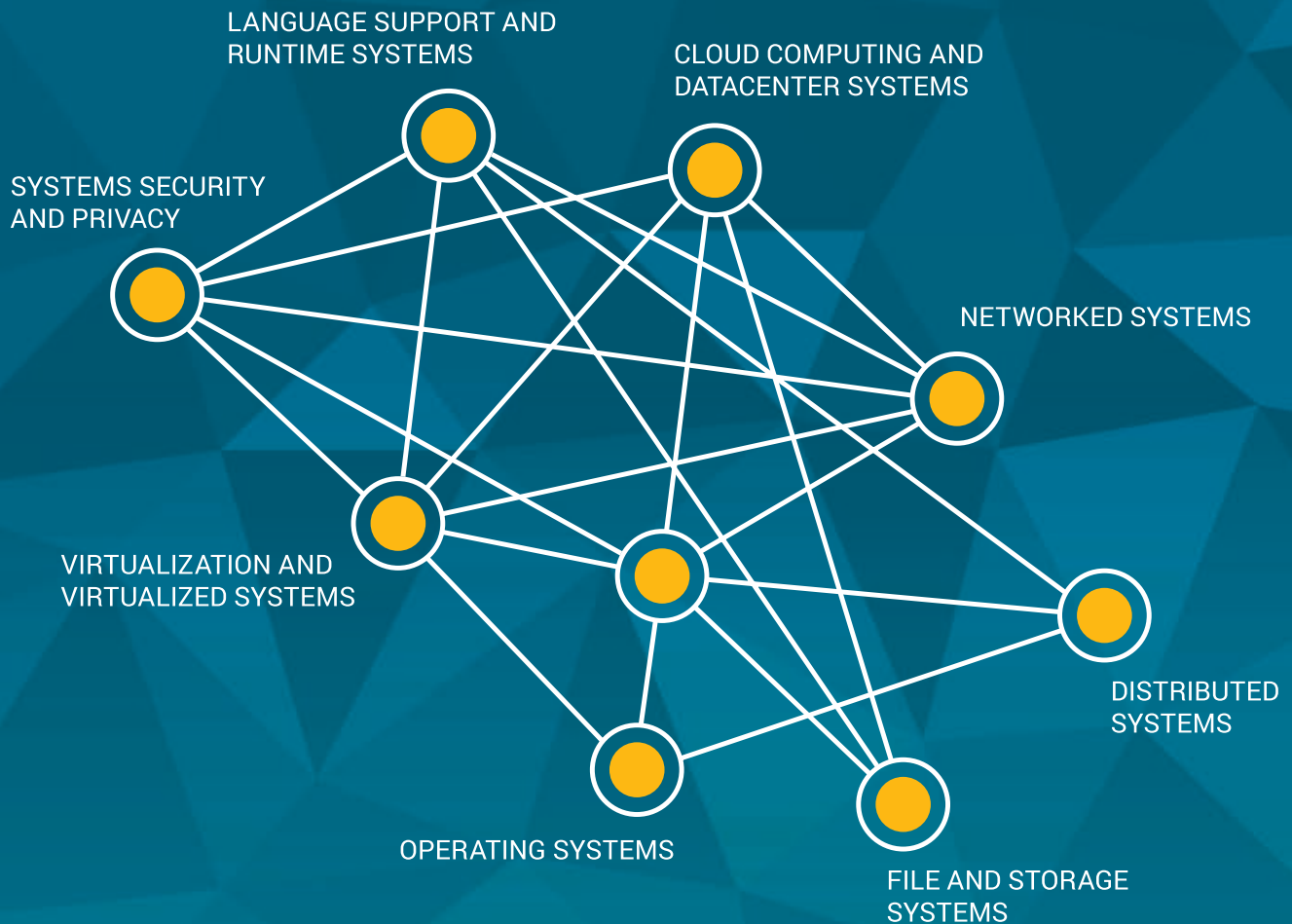
Sponsored by



27–30 APRIL 2020 | HERAKLION | CRETE | GREECE

15TH EUROPEAN CONFERENCE ON COMPUTER SYSTEMS

<EURO/SYS'20>



ORGANIZING COMMITTEE

GENERAL CHAIRS

Angelos Bilas (University of Crete, FORTH)
Kostas Magoutis (University of Crete, FORTH)
Evangelos Markatos (University of Crete, FORTH)

PROGRAM COMMITTEE CHAIRS

Dejan Kostic (KTH Royal Institute of Technology)
Margo Seltzer (University of British Columbia)

REGISTRATION OPEN

Early Bird Registration until 20 March 2020



For more information visit:

www.eurosys2020.org

eurosys2020@eurosys2020.org

#EuroSys20



Association for
Computing Machinery



SIGOPS



ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Departments

- 5 **From the President**
Engaging Future
Generations of ACM Leaders
By Cherri M. Pancake

- 7 **Cerf's Up**
A Hands-Free Ride
By Vinton G. Cerf

- 9 **Letters to the Editor**
Online Voting Still
Security Pipedream

- 9 **Calendar**

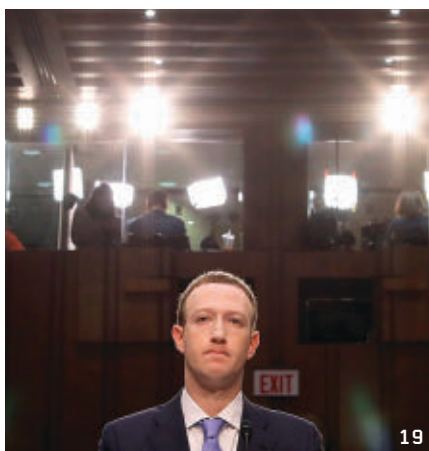
- 10 **BLOG@CACM**
Getting High School, College
Students Interested in CS
 Mark Guzdial considers how
 few U.S. high school students
 take computer science, while
 Robin K. Hill shares what she's
 learned in teaching first-year
 college computing students.

- 88 **Careers**

Last Byte

- 96 **Q&A**
RISCy Beginnings
 In a career launched by groundbreaking
 research, Garth Gibson continues to
 shepherd technological advances
 "from blackboard through standards
 and to commercial reality."
By Leah Hoffmann

News



- 13 **Malevolent Machine Learning**
 AI attacks throw light on
 the nature of deep learning.
By Chris Edwards
- 16 **Robots Aim to Boost**
Astronaut Efficiency
 A multitude of robotic assistants
 for astronauts and rovers are
 in development to make space
 exploration more resource-efficient.
By Paul Marks

- 19 **Regulating Information Technology**
 Why isn't IT regulated,
 when it can have such substantial
 impacts on people's lives?
By Keith Kirkpatrick

Viewpoints



- 22 **Computing Ethics**
Should Researchers Use Data
from Security Breaches?
 Evaluating the arguments for
 and against using digital data
 derived from security breaches.
By David M. Douglas
- 25 **Code Vicious**
Koding Academies
 A low-risk path to becoming
 a front-end plumber.
By George V. Neville-Neil
- 26 **The Profession of IT**
Uncertainty
 Considering how to best navigate
 stability and randomness.
By Peter J. Denning and Ted G. Lewis
- 29 **Viewpoint**
Public Entrepreneurship
and Policy Engineering
 Training the next generation
 of leader and problem solver.
By Beth Simone Noveck



Association for Computing Machinery
 Advancing Computing as a Science & Profession

Practice



32

- 32 Hack for Hire**
Investigating the emerging black market of retail email account hacking services.
By Ariana Mirian

- 38 API Practices If You Hate Your Customers**
Application programming interfaces speak louder than words.
By Thomas A. Limoncelli

q Articles' development led by acmqueue.queue.acm.org

Contributed Articles

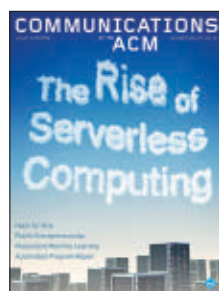


44

- 44 The Rise of Serverless Computing**
The server is dead, long live the server.
By Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski



Watch the co-organizers discuss this section in the exclusive *Communications* video.
<https://cacm.acm.org/videos/the-rise-of-serverless-computing>



About the Cover:
Serverless computing is a new paradigm for implementing applications that tap deep into the early promises of cloud computing. This month's cover story presents an overview of the current state of and future industry and research trends in serverless computing. Cover illustration by Peter Bollinger.

Review Articles

- 56 Automated Program Repair**
Automated program repair can relieve programmers from the burden of manually fixing the ever-increasing number of programming mistakes.
By Claire Le Goues, Michael Pradel, and Abhik Roychoudhury



Watch the co-organizers discuss this section in the exclusive *Communications* video.
<https://cacm.acm.org/videos/automated-program-repair>

- 66 Rethinking Search Engines and Recommendation Systems: A Game Theoretic Perspective**
Novel approaches draw on the strength of game theoretic mechanism design.
By Moshe Tennenholtz and Oren Kurland

Research Highlights

- 78 Technical Perspective**
Bootstrapping a Future of Open Source, Specialized Hardware
By Michael B. Taylor
- 79 OpenPiton: An Open Source Hardware Platform For Your Research**
By Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahradd, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO

Vicki L. Hanson

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Darren Ramdin

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Scott E. Delman

ACM COUNCIL

President

Cherri M. Pancake

Vice-President

Elizabeth Churchill

Secretary/Treasurer

Yannis Ioannidis

Past President

Alexander L. Wolf

Chair, SGB Board

Jeff Jortner

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Gabriele Kotsis; Susan Dumais; Renée McCauley; Claudia Bauzer Medeiros; Elizabeth D. Mynatt; Pamela Samuelson; Theo Schlossnagle; Eugene H. Spafford
SGB Council Representatives
 Sarita Adve and Jeanna Neefe Matthews

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

Terry Coatta

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Chris Hankin

ACM India Council

Abhiram Ranade

ACM China Council

Wenguan Chen

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson and Joseph Konstan

Board Members

Phoebe Ayers; Chris Hankin; Mike Heroux; Nenad Medvidovic; Tulika Mitra; Michael L. Nelson; Sharon Oviatt; Eugene H. Spafford; Stephen N. Spencer; Divesh Srivastava; Robert Walker; Julie R. Williamson

ACM U.S. Technology Policy Office

Adam Eisgrau,

Director of Global Policy and Public Affairs
 1701 Pennsylvania Ave NW, Suite 200,
 Washington, DC 20006 USA
 T (202) 580-6555; acmpo@acm.org

Computer Science Teachers Association

Jake Baskin

Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS

Scott E. Delman

cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Lawrence M. Fisher

Web Editor

David Roman

Editorial Assistant

Danbi Yu

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Bernadette Shade

Intellectual Property Rights Coordinator

Barbara Ryan

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

David Anderson; Michael Cusumano;

Peter J. Denning; Mark Guzdial;

Thomas Haigh; Leah Hoffmann; Mari Sako;

Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission

permissions@hq.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhlp@acm.org

Letters to the Editor

letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

WEB BOARD

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

AUTHOR GUIDELINES

http://cacm.acm.org/about-communications/author-center

ACM ADVERTISING DEPARTMENT

1601 Broadway, 10th Floor
 New York, NY 10019-7434 USA
 T (212) 626-0686
 F (212) 869-0481

Advertising Sales Account Manager

Ilia Rodriguez
 ilia.rodriguez@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

1601 Broadway, 10th Floor
 New York, NY 10019-7434 USA
 T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Andrew A. Chien

eic@cacm.acm.org

Deputy to the Editor-in-Chief

Lihan Chen

cacm.deputy.to.eic@gmail.com

SENIOR EDITOR

Moshe Y. Vardi

NEWS

Co-Chairs

Marc Snir and Alain Chesnais

Board Members

Monica Divitini; Mei Kobayashi;
 Rajeev Rastogi; François Sillion

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;

John Leslie King; Paul Rosenbloom

Board Members

Terry Benzel; Michael L. Best; Judith Bishop;

Lorrie Cranor; Boi Falting; James Grimmelman;

Mark Guzdial; Haym B. Hirsch;

Richard Ladner; Carl Landwehr; Beng Chin Ooi;

Francesca Rossi; Len Shustek; Loren Terveen;

Marshall Van Alstyne; Jeannette Wing;

Susan J. Winter

Q PRACTICE

Co-Chairs

Stephen Bourne and Theo Schlossnagle

Board Members

Eric Allman; Samy Bahra; Peter Bailis;

Betsy Beyer; Terry Coatta; Stuart Feldman;

Nicole Forsgren; Camille Fournier;

Jessie Frazelle; Benjamin Fried; Tom Killalea;

Tom Limoncelli; Kate Matsudaira;

Marshall Kirk McKusick; Erik Meijer;

George Neville-Neil; Jim Waldo;

Meredith Whittaker

CONTRIBUTED ARTICLES

Co-Chairs

James Larus and Gail Murphy

Board Members

William Aiello; Robert Austin; Kim Bruce;

Alan Bundy; Peter Buneman; Jeff Chase;

Andrew W. Cross; Yannis Ioannidis;

Gal A. Kaminka; Ben C. Lee; Igor Markov;

Lionel M. Ni; Adrian Perrig; Doina Precup;

Marie-Christine Rousset; Shankar Sastry;

m.c. schraefel; Ron Shamir; Sebastian Uchitel;

Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros, Shriram Krishnamurthi,

and Orna Kupferman

Board Members

Martin Abadi; Amr El Abbadi;

Animashree Anandkumar; Sanjeev Arora;

Michael Backes; Maria-Florina Balcan;

David Brooks; Stuart K. Card; Jon Crowcroft;

Alexei Efros; Bryan Ford; Alon Halevy;

Gernot Heiser; Takeo Igarashi;

Srinivasan Keshav; Sven Koenig;

Ran Libeskind-Hadas; Karen Liu; Greg Morrisett;

Tim Roughgarden; Guy Steele, Jr.;

Robert Williamson; Margaret H. Wright;

Nicolai Zeldovich; Andreas Zeller

SPECIAL SECTIONS

Co-Chairs

Sriram Rajamani, Jakob Rehof,

and Haibo Chen

Board Members

Tao Xie; Kenjiro Taura; David Padua

ACM Copyright Notice

Copyright © 2019 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhlp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
 1601 Broadway, 10th Floor
 New York, NY 10019-7434 USA

Printed in the USA.



Association for
Computing Machinery





DOI:10.1145/3368441

Cherri M. Pancake

Engaging Future Generations of ACM Leaders

ATTRACTING YOUNGER, NEXT-GENERATION members is a challenge faced by professional societies worldwide. ACM recognized early on that as a volunteer-driven organization, our future health depends not only on appealing to a diverse membership, but also on creating a pipeline of future leaders. We are in the process of developing new strategies to motivate recent graduates to get involved with ACM. In the meantime, I'd like to tell you about two programs that have already struck a chord with early-career professionals. Both were designed to offer young technologists and researchers the opportunity to network with peers from around the globe and to hone their leadership skills through personal interactions with more experienced leaders.

The **ACM Future of Computing Academy (FCA)** was created in 2016 to help shape the future of our organization. It brings together talented young professionals from a broad range of computing subdisciplines and institutions—academia, research labs, established companies, and startups—who want to perform valuable services for the community while expanding their professional networks and leadership experience. FCA members identify and implement pilot projects that address challenging issues facing our organization and the computing field in general. They work closely with ACM leadership, bringing new ideas/perspectives and helping to integrate successful pilots into ACM's institutional practices.

FCA members are selected biennially through a highly competitive process similar to those used for ACM awards. This summer's recruitment attracted hundreds of applications from around the globe, resulting in 36 new members.



1988 ACM A.M. Turing Award recipient Ivan Sutherland (fourth from right) visits with young researchers at the Heidelberg Laureate Forum last September.

The expanded membership represents 17 countries on all six inhabited continents. It is an extremely diverse group, in terms not just of gender and geography but also type of organization and disciplinary interests. Learn more about the FCA and its activities at <https://www.acm.org/fca>.

The **Heidelberg Laureate Forum (HLF)** is an annual event where young researchers in computing or mathematics share the scientific stage with world leaders who have been recognized with the ACM A.M. Turing Award, ACM Prize in Computing, Abel Prize, Fields Medal, or Nevanlinna Prize. ACM partnered with other leading scientific organizations to launch HLF in 2013. The event provides a unique opportunity for young researchers to interact with some of our industry's greatest innovators. HLF's week-long program is a carefully crafted mix of research updates, panels on controversial issues in the computing and math fields, group discussions, and informal social activities.

Approximately 200 young researchers are accepted each year for HLF, which takes place in Heidelberg (Germany) at the end of September. As with FCA, the application process is extremely competitive. Participants include undergraduates, postgraduate students, and post-

docs in approximately equal proportion. The 2019 event hosted 202 young researchers from 60 nations, 120 male and 82 female, many of them with interdisciplinary backgrounds. I strongly encourage young researchers to apply for next year's event. As you read this, the application portal has opened for the 8th Annual HLF, slated for September 2020 (<https://application.heidelberg-laureate-forum.org/>; the deadline is Feb. 14, 2020).

As a professional society, ACM leadership recognizes that the key to our long-term sustainability is to engage the ideas, energy, and enthusiasm of new generations. We must create opportunities for young professionals to grow their careers and give back to their communities through ACM. The ACM Future of Computing Academy and Heidelberg Laureate Forum are two examples where we have already seen success. But the journey has just begun. I encourage each of you to join in and help us explore the future of ACM. Share your own ideas with us about how to serve our growing community in new and forward-thinking ways. 

Cherri M. Pancake is President of ACM, professor emeritus of electrical engineering and computer science, and director of a research center at Oregon State University, Corvallis, OR, USA.

Copyright held by author/owner.

Inviting Young Scientists



Association for
Computing Machinery

Meet Great Minds in Computer Science and Mathematics

As one of the founding organizations of the Heidelberg Laureate Forum <http://www.heidelberg-laureate-forum.org/>, ACM invites young computer science and mathematics researchers to meet some of the preeminent scientists in their field. These may be the very pioneering researchers who sparked your passion for research in computer science and/or mathematics.

These laureates include recipients of the ACM A.M. Turing Award, the ACM Prize in Computing, Abel Prize, the Fields Medal, and the Nevanlinna Prize.

The 8th Heidelberg Laureate Forum will take place **September 20–25, 2020** in Heidelberg, Germany.

This week-long event features presentations, workshops, panel discussions, and social events focusing on scientific inspiration and exchange among laureates and young scientists.

Who can participate?

Students of computer science and mathematics (or closely related fields) at the Undergraduate/Pre-Master, Graduate Ph.D. and Postdoc levels can apply for the Heidelberg Laureate Forum. Postdocs include young researchers in classical postdoc positions as well as those who have recently completed (within five years) their Ph.D. and are still strongly interested in scientific matters, even if currently working in a non-scientific environment.

How to apply:

Online: <https://application.heidelberg-laureate-forum.org/>
Materials to complete applications are listed on the site.

What is the schedule?

The application deadline is **February 14, 2020**.

Successful applicants will be notified by **mid-late April 2020**.

More information available on
<https://www.heidelberg-laureate-forum.org/young-researchers/faq.html>





DOI:10.1145/3369587

Vinton G. Cerf

A Hands-Free Ride

I recently had the opportunity to take a ride in a Waymo self-driving car in Chandler, AZ. I had been looking forward to this experience, not only to see how well the technology worked

but also what the experience might be like as a passenger. Upon my arrival at the Waymo facility, I had apparently approached the side of the building where the Waymo cars go at the end of their duty cycles to be refueled and inspected. As I drove in, I was more or less surrounded by incoming Waymo vehicles. I relaxed as they navigated their way around me.

I was not unaccompanied on this test drive. There was an emergency backup driver at the wheel who did not need to take any actions during the 15-minute drive. In addition, I had an engineer with me who was able to show what the Waymo car's "brain" (my term) was seeing on the road. I also had a business operations escort who was able to answer questions about how this self-driving ride service was being rolled out. The rider in the back seat has a display showing the traffic nearby and providing information, alerts, opportunity to speak with a customer service person, and so on. Knowing what the car "sees" is reassuring.

Our merry band of four took off for a residential neighborhood. There were a number of challenging situations to deal with. One that impressed me mightily was a left turn at a two-way intersection with no traffic signal. It was a fairly busy road and I would have found it challenging to estimate when it was safe to turn had I been at the wheel. A second interesting maneuver involved a left turn at a traffic signal that included both a red light (to prevent traffic from advancing into the

intersection to cross the street) and a flashing yellow arrow that meant "turn left when it is safe to do so, proceed with caution." Both of these turns were made successfully.

In earlier experiences, I felt the car tended to ride somewhat unevenly as if the "driver" were prone to lightly tapping the brake. This ride seemed much smoother and more natural. We cruised around the residential neighborhood, negotiating a traffic circle, four-way intersections, cross traffic, pedestrians, a school area, and speed bumps. Our Waymo car negotiated all of these situations without apparent difficulty. I learned the cars have access to an extremely detailed amount of information about

We cruised around the residential neighborhood, negotiating a traffic circle, four-way intersections, cross traffic, pedestrians, a school area, and speed bumps.

the area in which they are operating so the sensor data obtained in real time is combined with known details of local conditions (such as speed bumps) to aid in decision making.

The car I was in displayed speed in km/hour but I didn't know that and momentarily panicked when I thought we were exceeding the posted mph speed limit! As my vehicle continued on its way, negotiating street crossings and turns, I marveled at the natural way it navigated in normal city traffic. The science of autonomous driving has come a very long way from the early grand challenge posed 15 years ago by the Defense Advanced Research Projects Agency. From those early beginnings, Waymo cars have driven millions of miles in city traffic and billions of miles in simulation.

Looking back on this experience, my sense of assurance that these cars really are equipped to handle unusual or at least complex traffic situations rose significantly. The level of care for safety that Waymo has taken has been documented in its reports to the U.S. Department of Transportation and the National Transportation Safety Board. This personal experience on the ground (err, road) reinforced my belief that self-driving car service is demonstrably feasible, especially in areas where weather conditions are favorable to its operation. 

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

Nominees for ACM's 2020 General Election

In accordance with the Constitution and Bylaws of the ACM, the Nominating Committee hereby submits the following slate of nominees for ACM's officers. In addition to the officers of the ACM, five Members at Large will be elected. The names of the candidates for each office are presented in random order below:

President (1 July 2020 – 30 June 2022):

Elizabeth Churchill, Google

Gabriele Kotsis, Johannes Kepler University Linz

Vice President (1 July 2020 – 30 June 2022):

Yannis Ioannidis, "Athena" Research Center and University of Athens

Joan Feigenbaum, Yale University and Amazon Web Services, Inc.

Secretary/Treasurer (1 July 2020 – 30 June 2022):

Jeff Jortner, Sandia National Laboratories

Elisa Bertino, Purdue University

Members at Large (1 July 2020 – 30 June 2024):

Alfred Spector, Two Sigma

John C.S. Lui, The Chinese University of Hong Kong (CUHK)

Tom Crick, Swansea University

Susan Dumais, Microsoft

Sanjiva Prasad, IIT Delhi

Mehran Sahami, Stanford University

Alejandro Saucedo, The Institute for Ethical AI & Machine Learning

Nancy M. Amato, University of Illinois Urbana-Champaign

The Constitution and Bylaws provide that candidates for elected offices of the ACM may also be nominated by petition of one percent of the Members who as of **1 November 2019** are eligible to vote for the nominee. Such petitions must be accompanied by a written declaration that the nominee is willing to stand for election. The number of Member signatures required for the offices of President, Vice President, Secretary/Treasurer, and Members at Large, is **739**.

The Bylaws provide that such petitions must reach the Elections Committee before **31 January 2020**. Original petitions for ACM offices are to be submitted to the ACM Elections Committee, c/o Pat Ryan, COO, ACM Headquarters, 1601 Broadway, 10th Floor, New York, NY 10019, USA, by **31 January 2020**. Duplicate copies of the petitions should also be sent to the Chair of the Elections Committee, Gerry Segal, c/o ACM Headquarters. Statements and biographical sketches of all candidates will appear in the May 2020 issue of *Communications of the ACM*.

The Nominating Committee would like to thank all those who helped us with their suggestions and advice.

Alexander L. Wolf, CHAIR,

Sarita Adve, **Chris Hankin**, **Abhiram Ranade**, **Chris Stephenson**



DOI:10.1145/3369867

Online Voting Still Security Pipedream

THE VIEWPOINT IN the September 2019 issue “Online Voting: We Can Do It! (We Have To),” while interesting, was flawed and failed to justify the claims made.

First and foremost, adoption targets are sensible only for mature, reliable technologies that solve clear public problems and where the harms, such as they are, can be mitigated. This is not true for online voting. As Hilarie Orman concedes, we don’t know how to build secure online voting systems. And as the critical iOS flaw and state-level attacks recently discovered by Google illustrate, critical security holes can linger unknown for years even on well-tested platforms.

Second, the technical measures proposed do not present a solution to the unique needs of elections. Voting is a two-party semi-anonymous transaction. Voters must be able to vote without revealing how they voted (known as the Australian Ballot) lest they be subject to intimidation or vote selling (such as beatings or bribes). Elections must be secure en masse lest honest votes be drowned by dishonest ones. Any system that allows outsiders to connect a voter to a vote, or to intercept and alter votes, gives that up. This is what allowed the man in the middle attack that took place in North Carolina.

This is fundamentally different from online shopping where the shopper, the vendor, and the financial service are each known, and transactions are independent. While the technologies involved are opaque, users can almost always follow the money. Despite this, online fraud still costs companies and individuals millions each year. Ultimately, online transactions are not so much secure as they are insured by parties who make enough to cover their losses. That is not possible with voting.

Third, it makes sense to adopt a fundamentally conservative view of election technology. The decisions of elections officials, decisions the author unfairly

dismisses as “draconian, discriminatory, and unsafe” are often driven by legal mandates and hard-won experience that have been crafted over decades. Relevant laws in most jurisdictions mandate public access. If anything, it is the rise of e-voting that has hidden the process from the general public. This is why many good government groups back paper ballots with electronic tabulation. Paper, when properly secured with a clear chain of custody, provides a public, machine-independent check on otherwise opaque systems.

Fourth, the column argues for Internet voting based upon the assumption that it would increase turnout. However, there is little guarantee of that. Indeed, one recent study of Internet voting by Micha Germann and Uwe Serdült entitled “Internet Voting and Turnout: Evidence from Switzerland” concluded that Internet voting did not increase turnout over traditional methods.

Elections are an essential public process. As David Eckhardt of Carnegie Mellon University has noted, voters must have a clear method to publicly verify that things are working as they should. Absent that, the system is both illegitimate and insecure. Any model that cannot provide that is unacceptable, and the premise that we will figure it out over time is never good enough.

Collin F. Lynch, Raleigh, NC, USA

Author’s response

Access to voting is a fundamental democratic principle. The use of mail-in ballots sent to all voters increases turnout, showing that in-person voting inhibits participation. Moreover, mail-in ballots reduce the cost of holding an election. These factors are even more favorable for Internet voting. Any technology that increases convenience and cost savings consistently upsets established norms, and that will be the case with voting.

Hilarie Orman, Woodland Hills, UT, USA

© 2019 ACM 0001-0782/19/12 \$15.00

Calendar of Events

Dec. 8–13

Middleware ‘19: 20th Int’l Middleware Conference, Davis, CA, Sponsored: ACM/SIG, Contact: Mohammad Sadoghi, Email: msadoghi@ucdavis.edu

Dec. 9–12

CoNEXT ‘19: The 15th Int’l Conference on emerging Networking EXperiments and Technology, Orlando, FL, Sponsored: ACM/SIG, Contact: Abedelaziz Mohaisen, Email: amohaisen@gmail.com

Dec. 15–18

MMAsia’19: ACM Multimedia Asia, Beijing, China, Sponsored: ACM/SIG, Contact: Changsheng Xu, Email: csxu@nlpr.ia.ac.cn

January 2020

Jan. 4–8

GROUP ‘20: The ACM Conference on Social Computing and Group Work, Sanibel Island, FL, Sponsored: ACM/SIG, Contact: Louise Barkhus, Email: barkhus@itu.dk

Jan. 19–25

POPL ‘20: The 47th Annual ACM SIGPLAN Symposium on Principles of Programming Languages, New Orleans, LA, Sponsored: ACM/SIG, Contact: Brigitte Pientka, Email: bpientka@cs.mcgill.ca

February

Feb. 4–6

FAT* ‘20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, Sponsored: ACM/SIG, Contact: Carlos A. Castillo, Email: chato@chato.cl

Feb. 9–12

TEI ‘20: 14th Int’l Conference on Tangible, Embedded, and Embodied Interaction, Sydney NSW, Australia, Sponsored: ACM/SIG, Contact: Lian Loke, Email: lian.loke@sydney.edu.au

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3365581

<http://cacm.acm.org/blogs/blog-cacm>

Getting High School, College Students Interested in CS

Mark Guzdial considers how few U.S. high school students take computer science, while Robin K. Hill shares what she's learned in teaching first-year college computing students.



Mark Guzdial
A Minuscule Percentage of Students Take High School Computer Science in the United States:

Access Isn't Enough

September 2, 2019

<http://bit.ly/2oupbk6>

If I told you only 4% of all high school students in the U.S. were taking science or math classes, you'd be aghast. That's such a tiny percentage! If 96% of students were *not* getting science or math classes, you could reasonably argue it does not exist in any practical sense.

Over the last few months, several reports provided new insights about U.S. high school computer science (CS):

- In California, only 3% of 1.9 million high school students enrolled in CS in 2017 (<http://bit.ly/2pq5kTH>).

- In Texas, 3.76% of its students completed a CS course in 2017–2018 (<http://bit.ly/2ox7cK6>).

California and Texas are two largest states based on U.S. population, but we can't generalize to *everyone* based on

those states. We don't have data on who is taking CS across the U.S., due to our state-centric, decentralized model of primary and secondary school education.

California and Texas are among the *leaders* in implementing CS education. Both are members of the Expanding Computing Education Pathways (ECEP) Alliance (<https://ecepalliance.org/>). Reports from ECEP states tell a similar story. For example, in Indiana, the most popular high school CS course enrolled only 2% of students in the state (<http://bit.ly/2oAq6Q6>). Miranda Parker is finishing her dissertation on factors influencing CS adoption at the high school level, and she's found only 1% of high school students in Georgia take a CS course. I suspect states not involved in ECEP have *lower* participation in CS, not higher. Given these numbers, it would be hard to believe more than 4% of U.S. high school students take a CS course.

One surprise for me in these reports: many U.S. high schools offer CS. I wrote a BLOG@CACM post in 2012 (<http://bit.ly/2ox5NDI>) estimating about 10% of high schools in the U.S. offered comput-

er science. Today, that number is much larger. In Texas, it is 43%. In California, it is 39%. In Georgia, it is around 40%, and about 1/3 of high schools in Indiana offer CS classes. The high schools that do offer CS are most often in larger (wealthier) high schools. Many students *could* take those classes. But they do not.

A decade ago, I thought the reason so few students were taking CS was access. Today's data paints a different story.

Even if CS is in the high school, few students sign up for it. Those who do are mostly male (71% in California, 74% in Texas). Some states are better at getting underrepresented minorities into CS classes. It's 48% in Texas. In California, only 16% of AP CS A exam takers are members of underrepresented groups.

Some states might decide this is not a problem; 3% may be enough. U.S. universities still reeling from overwhelming numbers of students in CS classes (<http://bit.ly/2oCXoOs>) might not *want* more students to become interested in CS at the high school level. I hear from schools that *most* introductory CS students they get have already had high school CS. These are compatible numbers. There are 15.3 million high school students in the U.S.; 3% of that is around 450,000 students—which I expect is more than the number of students in university intro CS in the U.S. If CS participation numbers increased at the high school level, probably more students would want to take CS at the university level (even if not as a CS major). Universities would have to restructure to manage such an enormous load.

Both the U.K. Computing at School

movement (<http://bit.ly/2oGcQJF>) and the U.S. CS for All movement (<http://bit.ly/2n4WCJS>) argue everyone deserves to learn about CS. If you buy their argument, these are disturbing numbers. We need a different strategy than simply getting CS into high schools. Now, we have to get students into the classes.

Comments:

South Carolina recently revised its high school computer science standards to require computer science beyond just keyboarding.

<http://bit.ly/2oDOMZG>

James Burton



Robin K. Hill Lessons from a First-Year Seminar

July 27, 2019

<http://bit.ly/2mZZuYm>

I teach a first-year seminar called “The Beauty and Joy of Computing” (after a course at Berkeley).³ All freshmen are required to take one of these seminars, offered in different subjects; the general pedagogical goal is teaching them to be college students. I taught it a year ago, and have a few reflections in preparation for a new round.

I am proud to work at a land-grant university with almost open enrollment, which requires careful treatment of disparate student backgrounds. The challenge presented by the computer science (CS) version of this course is that student interest in computing ranges from nil to avid, and student experience with computing ranges from none to expertise (the expertise ranges from narrow to haphazard). I do not pretend to have resolved these into a harmonious program of study, but I can sketch some of my attempts.

To introduce first-year students to the college environment, the course includes a research paper, group projects, and oral presentations. They need to learn to distinguish peer-reviewed results from public relations, and scholarship from argumentation and opinion. They need to be able to explain AI, Internet, and general hardware and software concerns to their peers, their families, and the public.

Major Components

Coding

I hold the pedagogical principle that in every class, students should come away

with a body of knowledge. In this case, it's a few basics of CS. We code or complete rudimentary Java programs with the BlueJ development environment. We look at number systems, character codes, encryption, compression, and similar topics up to the Fundamental Theorem of Arithmetic. Components that did not work for us include the block-style programming platform Snap!. I prefer beginners see plain source code in a text file, for the standard view of how computers are programmed.

Internet Issues

To engage students whose strengths lie in areas other than programming, we review contemporary Internet issues via the 2008 textbook by Abelson, Ledeen, and Lewis,¹ supplemented by journalism from *The New York Times* and other sources. A list of issues serves as the sign-up choices for the group project.

Literature

I believe young people interested in computers should also learn about the humanities (and so should other young people). I assign the reading of a classic novel that shares one of the themes we study. Last year, the country read *Frankenstein* in honor of its 200th anniversary. We joined in, discussing it as an account of the unintended consequences of technology,² notwithstanding that a 19th-century novel consisting mostly of lengthy soliloquies on guilt and misery is heavy going, with no actual technology.

Other classics that parallel issues of modern technology (besides 1984 and surveillance) include *The Scarlet Letter* and public shaming. Many great works of science fiction reflect tech issues, but I want a distant rather than close parallel, for purposes of abstracting the common theme. Suggestions are welcome!

General Pedagogy for Freshmen

Here are mixed tips for teachers, organized only by idiosyncrasy:

1. Don't cover up the difficult, tedious, fussy bits. Discuss input and output; variable declarations; defaults and parameters on the installation of software. These are counterintuitive protocols people must face to master computers.
2. Expose critical points of misunderstanding. Give exercises that distinguish

between $A[i]+1$ and $A[i+1]$; that interpret 11010011 as a char, number, and instruction; that contrast variable names such as *Temperature* and *temperature*.

3. If you mention UTF-8, or the Fundamental Theorem of Arithmetic, or a substitution cipher, tell them what it is right then and there. You can return to cover it comprehensively later.

Group project: To preclude students working with friends, have them form teams with classmates not already in their phones. Require meeting schedules and regular updates.

Research paper: Give incremental assignments, with suggestions incorporated into the next draft. Distinguish between objective journalism and research papers as acceptable sources, marketing and white papers as unacceptable.

I still seek suitable research publications. What journals or magazines provide peer-reviewed articles on CS for the neophyte? Our library subscribes to ACM and IEEE publications, but online. For browsing and discovery, I direct students to special technology sections or issues of major newspapers and magazines. Again, suggestions are welcome!

Teaching at the novice level invokes standard areas of philosophy:

- Metaphysics, in extracting the essence of the subject.
- Epistemology, in considerations of how to convey the essence of the subject.
- Ethics, insofar as the teacher answers the question of the subject, “What good is it?”
- Aesthetics, in how to make it attractive to learners. This may be a sly application rather than a scholarly endeavor.

Let's do our best for our students, whoever they are.

References

1. Abelson, H., Ledeen, K., and Lewis, H. *Blown to Bits: Your Life, Liberty, and Happiness after the Digital Explosion*. 2008.
2. Hill, R.K., “FictionStein,” BLOG@CACM, November 21, 2018, <http://bit.ly/2oEKyza>
3. The Beauty and Joy of Computing. University of California, Berkeley, Advanced Placement course for late high school and early college students. Accessed 25 July 2019.

Mark Guzdial is a professor in the Computer Science & Engineering Division of the University of Michigan, USA. Robin K. Hill is a lecturer in the Department of Computer Science and an affiliate of both the Department of Philosophy and Religious Studies and the Wyoming Institute for Humanities Research at the University of Wyoming, USA.

© 2019 ACM 0001-0782/19/12 \$15.00

SHAPE THE FUTURE OF COMPUTING. JOIN ACM TODAY.

www.acm.org/join/CAPP

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ☐ Professional Membership: \$99 USD
- ☐ Professional Membership plus
ACM Digital Library: \$198 USD
(\$99 dues + \$99 DL)

ACM STUDENT MEMBERSHIP:

- ☐ Student Membership: \$19 USD
- ☐ Student Membership plus ACM Digital Library: \$42 USD
- ☐ Student Membership plus Print *CACM* Magazine: \$42 USD
- ☐ Student Membership with ACM Digital Library plus
Print *CACM* Magazine: \$62 USD

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

PAYMENT INFORMATION

Name

Mailing Address

City/State/Province

ZIP/Postal Code/Country

- ☐ Please do not release my postal address to third parties

Email Address

- ☐ Yes, please send me ACM Announcements via email
☐ No, please do not send me ACM Announcements via email

- ☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Credit Card #

Exp. Date

Signature

Purposes of ACM

ACM is dedicated to:

- 1) Advancing the art, science, engineering, and application of information technology
- 2) Fostering the open interchange of information to serve both professionals and the public
- 3) Promoting the highest professional and ethics standards

By joining ACM, I agree to abide by ACM's Code of Ethics (www.acm.org/code-of-ethics) and ACM's Policy Against Harassment (www.acm.org/about-acm/policy-against-harassment).

I acknowledge ACM's Policy Against Harassment and agree that behavior such as the following will constitute grounds for actions against me:

- Abusive action directed at an individual, such as threats, intimidation, or bullying
- Racism, homophobia, or other behavior that discriminates against a group or class of people
- Sexual harassment of any kind, such as unwelcome sexual advances or words/actions of a sexual nature

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
Hours: 8:30AM - 4:30PM (US EST)

Fax: 212-944-1318
acmhelp@acm.org
www.acm.org/join/CAPP

Malevolent Machine Learning

AI attacks throw light on the nature of deep learning.

AT THE START of the decade, deep learning restored the reputation of artificial intelligence (AI) following years stuck in a technological winter. Within a few years of becoming computationally feasible, systems trained on thousands of labeled examples began to

exceed the performance of humans on specific tasks. One was able to decode road signs that had been rendered almost completely unreadable by the bleaching action of the sun, for example.

It just as quickly became apparent, however, that the same systems could just as easily be misled.

In 2013, Christian Szegedy and colleagues working at Google Brain found subtle pixel-level changes, imperceptible to a human, that extended across the image would lead to a bright yellow U.S. school bus being classified by a deep neural network (DNN) as an ostrich.

Two years later, Anh Nguyen, then a Ph.D. student at the University of



High-resolution images of fake "celebrities" generated by a Generative Adversarial Network using the CelebA-HQ training dataset.

Wyoming, and colleagues developed what they referred to as “evolved images.” Some were regular patterns with added noise; others looked like the static from an analog TV broadcast. Both were just abstract images to humans, but these evolved images would be classified by DNNs trained on conventional photographs as cheetahs, armadillos, motorcycles, and whatever else the system had been trained to recognize.

A 2017 attack centered on road-sign recognition demonstrated the potential vulnerability of self-driving cars and other robotic systems to attacks that took advantage of this unexpected property. In an experiment, a team from the University of Michigan, Ann Arbor demonstrated that brightly colored stickers attached to a stop sign could make a DNN register it as a 45mph speed-limit sign. This and similar attacks encouraged the U.S. Defense Advanced Research Projects Agency (DARPA) to launch a project at the beginning of this year to try to develop practical defenses against these attacks.

The key issue that has troubled deep learning researchers is why deep learning models seem to be fooled by what appears to humans like noise. Although experiments by James DiCarlo, a professor in neuroscience working at the Massachusetts Institute of Technology (MIT), and others have showed similarities between the gross structure of the visual cortexes of primates and DNNs, it has become clear the machine learning models make decisions based on information the brain either does not perceive, or simply ignores.

In work published earlier this year, the student-run Labsix group based at MIT found features recognized by DNNs can be classified into groups they call robust and non-robust.

Andrew Ilyas, Ph.D. student and Labsix member, says robust features are those that continue to deliver the correct results when the pixels they cover are changed by small amounts, as in Szegedy’s experiments. “For instance, even if you perturb a ‘floppy ear’ by a small pixel-wise perturbation, it is still indicative of the ‘dog’ class.”

Non-robust features, on the other hand, may be textures or fine details

Adversarial training provides deep neural networks with a series of examples that try to force the model to ignore features that have been shown to be vulnerable.

that can be disguised by lots of tiny changes to pixel intensity or color. “Imagine that there is a pattern that gives away the true class, but is very faint,” Ilyas suggests. It does not take much to hide it or change it to resemble a non-robust feature from a completely different class.

In work similar to that of the Labsix group, Haohan Wang and colleagues at Carnegie-Mellon University found that filtering out high-frequency information from images worsened the performance of DNNs they tested. Ilyas stresses that the work his group performed demonstrated that the subtle features are useful and representative, but they are easy to subvert, underlining, he says, “a fundamental misalignment between humans and machine-learning models.”

Researchers have proposed a battery of methods to try to defend against adversarial examples. Many have focused on the tendency of DNNs to home in on the more noise-like, non-robust features. However, attacks are not limited to those features, as various attempts at countermeasures have shown. In one case, a team of researchers working at the University of Maryland used a generative adversarial network (GAN) similar to those used to synthesize convincing pictures of celebrities. This GAN rebuilt source images without the high-frequency noise associated with most adversarial examples and was,

for a while, proved hard to fool. But eventually another team warped images using larger-scale changes to create adversarial examples that did beat Defense-GAN.

The most resilient approach so far is that of adversarial training. This technique provides the DNN with a series of examples during the training phase that try to force the model to ignore features that are shown to be vulnerable. It is a technique that comes with a cost: experiments have revealed that such training can just as easily hurt the performance of the DNN on normal test images; networks begin to lose their ability to generalize and classify new images correctly. They start overfitting to the training data.

“When training our model with adversarial training, we explicitly discourage it from relying on non-robust features. Thus, we are forcing it to ignore information in the input that would be useful for classification,” Ilyas notes. “One could argue, however, that the loss in accuracy is not necessarily a bad thing.”

Ilyas points out that the lower accuracy based on robust models is probably a more realistic estimate of a machine learning model’s performance if we are expecting DNNs to recognize images in the same way humans do. Ilyas says one aim of the Labsix work is to close the gap between human and machine by forcing the DNNs to home in on larger features. This will have the effect of making it easier for humans to interpret why the models make the mistakes they do.

However, with conventional DNN architectures, there is still some way to go to close the gap with humans, even if non-robust features are removed from the process. A team led by Jörn-Henrik Jacobsen, a post-doctoral researcher at the Vector Institute in Toronto, Canada, found it is possible for completely different images to lead to the same prediction. Not only that, adversarially trained DNNs that focus on robust features seem to be more susceptible to this problem.

A statistical analysis performed by Oscar Deniz, associate professor at the Universidad de Castilla-La Mancha in Spain, and his colleagues suggests a deeper issue with machine

learning models as they exist today that may call for architectural enhancements. Deniz says the presence of adversarial examples is a side-effect of a long-standing trade-off between accuracy and generalization: “From my point of view, the problem is not in the data, but in the current forms of machine learning.”

A different approach to combating adversarial examples that does not rely on changes to the learned models themselves is to find ways to determine whether a machine learning model has not gone further than its training should allow. A major problem with DNNs in the way they are constructed today is that they are overly confident in the decisions they make, whether rightly or wrongly. The chief culprit is the “softmax” layer used by most DNN implementations to determine the probability of the image being in any of the categories on which it was trained.

Nicolas Papernot, a research scientist at Google Brain, explains, “The softmax layer is a great tool for training the model because it creates a nice optimization landscape, but it is not a suitable model for making predictions. A softmax layer does not allow the model to refuse to make a prediction. It is not surprising, then, that once presented with an input that it should not classify, a neural network equipped with a softmax outputs an incorrect prediction.”

Originally developed by Papernot while he was a Ph.D. student together with Patrick McDaniel, professor of information and communications science at Pennsylvania State University, the Deep k-Nearest Neighbors (DkNN) technique performs a layer-by-layer analysis of the decisions made by the machine learning model during classification to construct a “credibility score.” Adversarial examples tend to lead to results that are not consistent with a single class, but with multiple different classes. It is only toward the end of the process that the softmax layer pushes up the probability of an incorrect result to a high-enough level to push the result off-target.

“The DkNN addresses the uncertainty that stems from learning from limited data, which is inevitable,”

Papernot says. The idea behind using DkNN to detect adversarial examples is to ensure the model makes a prediction only when it has enough training data to call upon to be able to generate a high-enough credibility score; otherwise, it will say it does not know, and a system relying on that DNN would either need to seek a second opinion, or to try to obtain more data.

Having developed attacks on DkNN together with his supervisor David Wagner, a professor of computer science at the University of California, Berkeley, Ph.D. student Chawin Sitawarin says an issue with the current approach is that it tends to suffer from false positives: correct classifications that have unusually low credibility scores. Sitawarin says improvements to the way the score is calculated could increase reliability, and that DkNN-like techniques represent a promising direction for detecting adversarial examples.

As work continues on multiple fronts, it seems likely that defense against these attacks will go hand-in-hand with greater understanding of how and why DNNs learn what they do. **C**

Further Reading

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial Examples Are Not Bugs, They Are Features
ArXiv preprint (2019): <https://arxiv.org/abs/1905.02175>

Wang, H., Wu, X., Yin, P., and Xing, E.P. High Frequency Component Helps Explain the Generalization of Convolutional Neural Networks
ArXiv preprint (2019): <https://arxiv.org/abs/1905.13545>

Papernot, N., and McDaniel P. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning
ArXiv preprint (2018): <https://arxiv.org/abs/1803.04765>

Jacobsen, J.H., Behrmann, J., Carlini N., Tramer, F., and Papernot, N. Exploiting Excessive Invariance caused by Norm-Bounded Adversarial Robustness
ICLR 2019 Workshop on Safe ML, New Orleans, Louisiana.
<https://arxiv.org/abs/1903.10484>

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

© 2019 ACM 0001-0782/19/12 \$15.00

ACM Member News

LOOKING AT WAYS TO SPEED UP THE INTERNET



Bruce Maggs,
Pelham Wilder
Professor of
Computer
Science at
Duke
University, first

became interested in computers in the mid-1970s. His father, a computer hobbyist, had installed a home computer system that had access to PLATO, a multiuser computing platform developed at the University of Illinois in the 1960s.

This early interest eventually led Maggs to earn his undergraduate, master's, and Ph.D. degrees in computer science from the Massachusetts Institute of Technology. After graduating, he spent time working for NEC Research Institute in Princeton, NJ, before joining the faculty of Carnegie Mellon University in 1994. He moved to Duke University in 2010.

In 1998, Maggs helped launch content delivery network Akamai Technologies, and served as its first vice president of research and development. He retains a part-time position at Akamai as vice president of research.

While his research focuses on distributed systems, including content delivery networks, computer networks, and computer and network security, lately Maggs has been concentrating on networking at the speed of light.

“I think the Internet is still too slow,” Maggs says. He explains he has been working on reducing latency to the point where a packet can be sent from point A to point B at the speed of light. In the future, Maggs hopes there will be a much broader class of applications (such as games, e-commerce deliveries, and virtual reality) able to take advantage of lower-latency networks.

“We will need the infrastructure and protocols in place so we can enjoy these latency-sensitive applications more than we can today.”

—John Delaney

Robots Aim to Boost Astronaut Efficiency

A multitude of robotic assistants for astronauts and rovers are in development to make space exploration more resource-efficient.

FROM FREE-FLYING DROIDS to humanoids, from crawlers to inflatable torsos, space robots of myriad types are now being considered for missions in low Earth orbit, on interplanetary spacecraft, and on other worlds.

It might sound like a prop list from a *Star Wars* movie, but space agencies and their contractors are developing a panoply of robotic assistants with a serious aim in mind: to boost the productivity and safety of astronauts.

The idea behind robot assistants is multifaceted: one aim is to offload time-consuming repetitive tasks like space station cleaning and inventory making from crew members to free-flying or humanoid robots. Ground robots controlled from, say, spacecraft orbiting the Moon or Mars could construct human habitats ahead of a landing, or perform reconnaissance ahead of human exploration missions.

In addition, the dangers of space junk, as well as the risks of cosmic radiation exposure and depressurization during spacewalks, could be quelled if a humanoid robot does the work, controlled by an exoskeleton-wearing astronaut more safely ensconced within a spacecraft.

At the heart of this drive for robotic assistance is the fact that robots need few of the quickly depleted resources astronauts burn up so very readily—principally oxygen, water, and power. The hope is that by taking on the drudge work of space, robots should be able to save spacefarers time, giving them the chance to focus on the parts of missions requiring human intelligence.

What kind of missions are being envisioned for robots? Terry Fong, director of the Intelligent Robotics Group at the NASA Ames Research Center near San Francisco, sees a signature example in a startling discovery made



NASA Expedition 60 flight engineer Christina Koch with the SPHERES robots on the International Space Station.

on the lunar surface in December 1972. “On Apollo 17’s second extravehicular activity (EVA), astronaut Jack Schmitt found orange volcanic glass at the Shorty crater. It was certainly one of, if not *the* most interesting, discoveries made by Apollo 17,” says Fong.

“But when they found it, they were three-quarters of the way through their excursion, and they had limited time because of their oxygen and power levels, and they had to get back to the lunar module,” he says. As a result, they had to rush their investigation.

However, if the Apollo 17 crew had been able to deploy some robotic reconnaissance to scout the area ahead of time, says Fong, “They could have changed the timing, they could have changed the route, they could have gone there first and spent more time there,” he says.

This should inform similar future missions, he says. “This would allow the humans to be a lot more efficient when they do go and do their work.”

NASA is not alone here. At the European Space Research and Technology Center (ESTEC), the research center for the European Space Agency (ESA) in The Netherlands, engineers are planning to develop ground robot assistants that could be teleoperated by astronauts orbiting a moon or planet or based in a space station like the ISS, or NASA’s planned Lunar Orbital Platform-Gateway. Such robots could perform the kind of reconnaissance Fong suggests, but equally, says Thomas Krueger, a robotics systems and software engineer in ESTEC’s Human-Robot Interaction Lab, they could be used to build and maintain infrastructure for a planetary habitat.

For the moment, however, such planetary surface robotic assistants are research projects, and the drive for greater robotically fueled efficiency in crewed spaceflight has begun at a much more modest level. It kicked off in May 2006 with the introduction to the International Space Station of

three free-flying robot assistants labeled Synchronized Position Hold, Engage, Reorient, Experimental Satellites (SPHERES).

The SPHERES are actually 18-sided polyhedrons just 21-cm across and with a mass of 4kg (8.8 lbs.). Functionally, at least, they resemble Luke Skywalker's light sabre training droid in the first *Star Wars* movie. Developed by the Space Systems Laboratory of the Massachusetts Institute of Technology (MIT), they use tiny puffs of CO₂ gas to dart around the ISS. The SPHERES have ultrasound sensors that receive inaudible chirps from ultrasound beacons placed around any module they work in, allowing the robots to compute their position in three-dimensional space.

In addition to carrying gadgets such as smartphones and cameras for astronaut experiments, the SPHERES are also a cheap satellite research platform allowing swarm flying and rendezvous and docking experiments to be performed inside the ISS. Crucially, the robots have provided the testbed for a more-powerful successor: another free-flying astronaut assistant robot called the Astrobee, developed by Fong's team at NASA Ames.

Having arrived on the ISS in April, the three Astrobees (dubbed Honey Bee, Bumble Bee, and Queen Bee) are cubes with 30cm-long sides. These droids' propulsion in zero gravity comes from electric fans, while their navigation is based on image recognition AI. "The Astrobees have louvred vanes that direct the air from the fans and so, in some senses, are more like drones," says Fong.

The Astrobees' image-based navigation, which uses cameras to create a dense three-dimensional feature map of space station module interiors, frees the ISS crew from having to set up indoor GPS beacons, as they did with SPHERES. "This is a critical feature as it allows Astrobees to be very flexible. It makes the whole world of the ISS its navigation system," says Fong.

What can Astrobees do? They each sport three payload bays on which useful tools can be plugged in, such as a robot arm, a novel sensing system, or a QR/barcode reader for inventory taking.

Fong says NASA has "explicitly gone out of their way" to allow third-party developers to invent for As-

Designers test their space robots on Earth before deployment, but must be sure that doing so in a non-representative environment does not breed false confidence.

trobee, as the data, mechanical, and electrical interfaces are openly published and the software is open source and based on the Robot Operating System (ROS). "It's an extensible open platform," he says.

While free flyers have their place on many tasks, however, in a space station replete with humans, some tasks will need human-shaped dexterous robots to work alongside people. However, this is a work in progress, and no humanoid robot is currently in regular operation on a spacecraft.

Way ahead in the research stakes, however, is the NASA Robonaut program that began in 1996 at the NASA Johnson Space Center in Houston, TX. NASA Johnson engineers introduced Robonaut 1 to the world in 2002: a humanoid torso with dexterous, gripping hands and soft, compliant arms that cannot hurt humans, along with a raft of sensors. That initial testbed did not fly to the ISS, but in 2011, Robonaut 2 was flown to the ISS aboard Space Shuttle *Discovery*, and was used in multiple dexterous robot tests there.

NASA says Robonaut 2 was "good at cleaning the many handrails inside the station," allowing astronauts to focus on key science and repair work. The droid, controlled by an astronaut in a VR helmet, also proved dexterous enough to become proficient at flipping switches and pushing buttons.

Some of those tests were conducted by Fong and his colleagues at NASA Ames. "Robonaut 2 was on the space station for several years and we learned

quite a bit about it. And then it had some issues with its power system and had to be brought back down to Earth," he says.

Says ESA's Krueger, "NASA had the Robonaut on board the station and it's a nice device but it is not really a system you can tell: 'hey, go to that module and give me that box'. We are not there yet. But it might be possible that we can control that robot better from the ground via remote operation."

More recently, the NASA Johnson Space Center has developed a newer humanoid space robot called R5, perhaps better known as Valkyrie, which builds on the experience with Robonaut 2. With new electronics, actuators, and sensors, it is now being further engineered for "break-throughs in humanoid control, motion planning, and perception" at the University of Edinburgh in the U.K., according to that institution's School of Informatics, which is working on the 125-kg, 1.8-meter-tall robot in a joint research program with nearby Heriot-Watt University.

Still another innovation in space-based humanoid assistants, and one receiving advanced concept seed funding from NASA, is the inflatable upper-torso robot being developed by Marc Killpack, a professor of mechanical engineering, and his colleagues at Brigham Young University. This robot addresses two of spaceflight's enduring problems: payload launch mass and storage space.

A robot whose torso and limbs can be inflated to a useful stiffness would have far less mass than a similarly sized robot with metallic arms and chest. When not in use, the droid can deflate its torso and limbs to take up less storage space.

Inflatable robots are not as crazy as they might sound; Fong's team previously worked with R&D contractor Otherlab on such a concept. "I'm actually quite excited about them," says Fong. "They have a much higher strength-to-weight ratio than traditionally designed rigid assemblies. That's because of the power of using fluid pressure. And the fabric-based manipulators, some of them can be much more intrinsically safe in terms of impact because they are very low inertia."

ESA's main focus is on controlling planetary surface robots from orbit,

which is why that agency has been refining the technology of robotic humanoid teleoperation by minimizing the effects of signal latency, says Krueger. In October 2018, for instance, ESA astronaut Alexander Gerst aboard the ISS remotely controlled a four-wheeled, twin-armed, humanoid robot named Rollin' Justin situated on a mocked-up Martian surface in the labs of the robot's maker, aerospace firm DLR in Oberpfaffenhofen, Germany. Using a point-and-click tablet-based teleoperation control app developed by ESA, Gerst successfully retrieved antenna parts and replaced a burned-out computer circuit using the robot, while orbiting the Earth at 28,000 kph.

While such tests have validated ESA's teleoperation technology, the question remains of how an actual space-qualified robot would perform on Mars. Rollin' Justin was built for testing on a smooth lab floor on Earth, with handy QR codes on the walls telling it where it is. When redesigned for the harsh space environment (with less-capable radiation-hardened electronics installed, and using image recognition for navigation), Krueger concedes its performance would be "downgraded."

This illustrates the profound issues faced by space robot designers: they must test their designs on Earth before deployment, but they also must ensure doing so in a non-representative environment does not breed false confidence. With zero gravity to contend with, plus cosmic radiation impacts flipping bits in memory chips and microprocessors, and the extreme heat and cold in space, there are many differences between Earth and space environments with which to cope.

"For instance, we work against gravity here on Earth, but in space that constant force vector is not there. A robot arm's gearbox or drive system has to counteract gravity on Earth, but not in space," says Krueger.

Zero-gravity aircraft trips constitute one option for testing in three dimensions. Another is an ultra-smooth granite table on which NASA tested the Astrobee: they placed each robot on an "air puck" which fires CO₂ downward to make the Astrobee float atop the smooth granite. "It gives you a really, really thin cushion of gas to float on. It's amazing; it's very close to being

frictionless when you push it around," says Fong.

What is the most debilitating computer engineering challenge for space robots? "One of the big challenges we have, and will have for a very long time, is that the gap between the capabilities of space and terrestrial computers continues to actually increase," says Fong. We may see strong progress here on Earth in self-driving cars and drones, he says, but the high-performance VLSI technology behind that does not reach the spaceflight community because it is not available in radiation-hardened form to protect it from the rigors of space.

Krueger agrees, "Space-grade processors cannot be as densely integrated as those used on Earth, and so are way slower, and that places computational limits on the image processing and machine learning techniques we can use in space applications," he says.

For instance, says Fong, the Rad-750, a common radiation-hardened 32-bit single-board computer for space applications, is based on the 1997 IBM/Motorola PowerPC chip, and its top clock speed of 200MHz runs two orders of magnitude slower than the technology in our smartphones. "That limits the algorithms we can use," Fong says.

Yet there are signs of hope. NASA's Mars 2020 Rover, Fong says, will utilize field programmable gate arrays (FPGAs) that allow new, smarter processing logic circuitry to be configured on the fly as new algorithms are developed (something Microsoft already does on its Azure cloud, allowing new cloud search algorithms to be uploaded without having to replace thousands of datacenter processors). The 2020 Rover also will deploy its own flying robotic assistant, called the Mars Helicopter, to map terrain on brief reconnaissance missions. Thanks to newer methods of tolerating radiation rather than blocking it completely, the Rover will use a modern Qualcomm Snapdragon processor.

Not all robotic solutions are going to be popular, however. Cosmic radiation, solar flares, space junk, and the risk of spacesuit depressurization are some of the things that make EVAs (extravehicular activity, or spacewalks) hazardous. Krueger recently found that the ESA's proposed robotic answer to these risks did not go down too

well. The agency's forthcoming Space Exoskeleton Controller (SPOC) will be worn by an ISS crewmember inside the station to control a humanoid robot out in space, perhaps swapping out ISS batteries or maintaining its solar arrays. Russia's space agency, Roscosmos, has similar plans.

However, Krueger says, "When I told an astronaut that with SPOC he would not need an EVA, he was not excited about it at all. In fact, he asked that we don't take the EVAs away from them."

Just like many people here on Earth who don't want machines taking their jobs, even those with the Right Stuff don't want to lose the most talismanic job in spaceflight to a robot. **C**

Further Reading

Apollo 17 Preliminary Science Report, National Aeronautics and Space Administration, 1973
<https://www.hq.nasa.gov/alsj/a17/as17psr.pdf>
 Astrobee Research Publications, National Aeronautics and Space Administration Ames Research Center, 2019
<https://www.nasa.gov/content/research-publications-0>

Bualat, M., Smith, T., Smith, E., Provencher, C., Fong, T., Smith, E.E., Wheeler, D.W., et al. Astrobee: A New Tool for ISS Operations, *Proceedings of AIAA SpaceOps*, Marseille, France.
<https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2517>

Smith, T., Barlow, J., Bualat, M., Fong, T., Provencher, C., Sanchez, H., Smith, E., et al. Astrobee: A New Platform For Free-Flying Robotics on the International Space Station, *Proceedings of 13th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*
<https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160007769.pdf>

Coltin, B., Fusco, J., Moratto, Z., Alexandrov, O., and Nakamura, R. Localization from Visual Landmarks on a Free-flying Robot, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*
<https://ntrs.nasa.gov/search.jsp?R=20160012286>

Hyatt, P., Kraus, D., Sherrod, V., Rupert, L., Day, N., and Killpack, M.D. Configuration Estimation for Accurate Position Control of Large-Scale Soft Robots, *IEEE/ASME Transactions on Mechatronics*.
<https://ieeexplore.ieee.org/document/8510913>

Paul Marks is a technology journalist, writer, and editor based in London, U.K.

Regulating Information Technology

Why isn't IT regulated, when it can have such substantial impacts on people's lives?

IN THE SPRING of 2018, Facebook CEO Mark Zuckerberg was called to testify before Congress, largely in response to the news that British consulting firm Cambridge Analytica had captured and used the data of more than 87 million Facebook users to influence elections without the social media giant's knowledge or consent, as well as the admission that Facebook itself had been used by Russians to spread fake news and propaganda. During his testimony, Zuckerberg laid out his views on both the need for, and inevitability of, regulation of technology companies and their products and services.

Zuckerberg has since published a call for governments to regulate the Internet by limiting harmful content, addressing long-standing privacy concerns, securing the integrity of elections, and ensuring data portability. However, as of the writing of this article, there has been little to no substantive action on the part of the U.S. federal government to address these and other IT-related concerns.

The lack of consistent and widespread regulation of Internet and computer technology, particularly around data privacy and security, within the U.S. points to a confluence of complicated factors that make the creation of new federal rules related to technology unlikely, despite the public's growing desire to have some entity police how their personal data and information is tracked, collected, and used.

"We're under the 20-some-year-old framework that doesn't really work closely for the industry," says Lee McKnight, an associate professor in the School of Information Studies at Syracuse University, referring to the Telecommunications Act of 1996, the last major piece of federal regulation that addresses the activities of com-



Facebook chairman Mark Zuckerberg was called to testify before U.S. Senate committees after it was reported U.K. consulting firm Cambridge Analytica had harvested the data of more than 87 million Facebook users without their consent to influence elections.

munication and IT companies. While the Act deregulated the communications business, letting any company offer local and long-distance telephony, cable TV programming, and other video services, its only reference to the issues of data privacy or security is Section 725's reference to prohibiting local exchange carriers from recording or using the "the occurrence or contents of calls received by providers of alarm monitoring services for the purposes of marketing such services on behalf of such local exchange carrier, or any other entity."

Indeed, McKnight says that, given the vast technological and business model changes that have oc-

curred over the past 23 years, the law is largely outdated and simply does not address how companies collect, share, monetize, and protect personal and service-usage data, nor does it address the issue of data security breaches. Regulatory activity advocates generally believe the collection of such a large amount of data, its huge commercial value, and the potentially negative repercussions of this data falling into the wrong hands portends the need for strong regulatory controls, along with stiff penalties for noncompliance.

Within the U.S., several government regulations have been enacted that address general cybersecurity issues, in-

cluding the Cybersecurity Information Sharing Act (CISA), the Cybersecurity Enhancement Act of 2014, the Federal Exchange Data Breach Notification Act of 2015, and the National Cybersecurity Protection Advancement Act of 2015. However, these regulations do not specifically address computer-related industries such as Internet service providers (ISPs) and software companies (or service companies such as social media sites), and many of the regulations include vague language that leaves a significant amount of room for interpretation.

Opponents of additional regulatory activity argue that the personal data that is most sensitive and in need of regulatory protection is already protected by regulations such as the Health Insurance Portability and Accountability Act (HIPAA), which provides for data privacy and security of individuals' medical information.

Furthermore, getting additional, specific data handling, privacy, and security regulations that will work in a global economy may be challenging to implement, as well as costly and some-

Getting additional, specific data handling, privacy, and security regulations that work in a global economy may be challenging to implement and costly to enforce.

what difficult to enforce. Many modern-day business services, such as social media sites, are global in nature, and it can be difficult to determine the applicable jurisdiction (for example, if a South African citizen purchases goods from a French-domiciled company via the Facebook platform, and the shipment of such goods comes from China, which data laws take precedence?).

Additionally, technology companies whose business models are primarily based around monetizing personal data, either directly for sales and marketing purposes, or indirectly by offering access to or selling that data to others, are generally not in favor of regulation that will prescribe how they collect, use, and store data. The argument most commonly summoned by regulatory opponents is that regulation tends to stifle innovation by limiting the creative ways in which data can be used, as well as by increasing compliance costs through requirements for enhanced and more frequent data updating, recordkeeping, and notification of data security breaches.

However, any regulation that limits how a business can capture and use personal data is likely to negatively impact a business that currently relies on relatively unfettered access to and use of that data.

Despite some opposition by technology companies, regulatory progress is being made. The European Union (EU) enacted in 2018 the Gen-

ACM News

Underwater Drones Make Waves

Understanding what lies beneath the surface of the sea has always been a formidable challenge. Visibility is terrible, moving can be difficult, and the ambient pressures become deadly with depth. Although submarines and modern equipment can peer beneath the surface, the world's oceans remain mysterious places that have not been fully explored.

That is beginning to change, however. Autonomous underwater vehicles (AUVs)—known as underwater drones—are quietly reshaping oceanographic research. “These robotic systems allow researchers to gather data in ways that were difficult or impossible in the past. They are force multipliers,” says Vicki Ferrini, a research scientist at the Lamont-Doherty Earth Observatory at Columbia University.

Equipped with sensors, sophisticated propulsion systems, and advanced software, underwater drones are helping researchers map the sea floor, understand currents, view dead

zones, conduct fish counts, improve tsunami prediction, and more. They also are aiding in finding shipwrecks, building windfarms, and accomplishing tasks including inspecting deep-sea oil platforms and nuclear reactors in vessels.

Robotic technology is reinventing underwater exploration. Only about 15% of the world's oceans have been mapped at a resolution of 100 meters—and even most of these areas remain largely unexplored. “Underwater drones provide continuous data about the sea floor and they deliver far greater resolution than we've had in the past,” Ferrini says.

Autonomous underwater drones are making their mark. A high-profile example is Seabed 2030, a joint effort of two nonprofits that aims to map the entire ocean floor in just over a decade.

“The first step in better protecting the oceans is to have a better understanding of them,” says Rochelle Wigley, project director for the Nippon Foundation GEBCO Project

Center for Coastal and Ocean Mapping/Joint Hydrographic Center of the University of New Hampshire.

Because underwater drones can't communicate as freely with the cloud and other computing systems for extended periods, on-board navigation and programming is critical, says Nathan Michael, director of Carnegie Mellon University's Resilient Intelligent Systems Lab. “These systems require a great deal of robotic perception in order to have the level of underwater autonomy needed.” The type and positioning of the sensors is critical for navigation when GPS and communications networks are not available. Drones also require specialized programming to find wrecks, spot specific types of sea life, and accomplish other specialized tasks.

Wigley says internal programming is a critical issue, and researchers are constantly looking for ways to build better algorithms, software models,

and avoidance systems. Ferrini believes further advances in computing power, improved battery systems, and more advanced sensors will unlock many of the secrets of oceans, deep-water lakes, and other underwater areas.

The use of these drones will undoubtedly grow in the coming years. In July, an underwater drone found the French submarine *Minerve*, which vanished off the southern coast of France in 1968. The wreckage was discovered at a depth of nearly 2,500 meters, or 1.5 miles. The U.S. Navy has also developed autonomous underwater vehicles it plans to use for a variety of purposes, including the waging of war.

Concludes Ferrini, “We're at a turning point where the on-board intelligence, sensing, navigation, and software programming are allowing underwater drones to tackle sophisticated tasks.”

—Samuel Greengard is an author and journalist based in West Linn, OR, USA.

eral Data Protection Regulation (GDPR), which regulates the handling and privacy of the data of individual citizens of the EU and the European Economic Area (EEA), and also addresses the export of personal data outside the EU and EEA regions. Because multinational companies operate both within the EU and beyond it, some companies are applying GDPR rules across all of their customer base, thereby providing indirect regulatory control to areas such as the U.S., which does not have a national regulatory framework covering data privacy or security.

“Many companies at this point would be fine complying with GDPR, with similar rules worldwide, and many companies, in fact, have done that,” says James Grimmelmann, a professor of law at Cornell Tech and Cornell Law School. However, given the restrictive nature of some of the components of GDPR, such as the requirement that individuals can request erasure of personal data related to them on any one of a number of grounds within 30 days of its publication, certain business models may not be compatible with GDPR.

“It may be that the kind of ad networks that Facebook and Google use simply can’t operate under GDPR,” Grimmelmann says, adding, “it will take a few years to find out whether that’s the case or not.”

While efforts to protect individuals’ privacy and personal data in the U.S. are under way, much of the work being done is at the local or regional level, rather than the federal level. For example, the California Consumer Privacy Act was signed by then-governor Jerry Brown last year and is set to go into effect in January. The Act conveys three main rights to consumers in California: the right to know what information is being collected about them; the ability to tell a business not to share or sell their personal information; and the right to have data protections put in place by companies that collect and store personal data. It should be noted, however, that Californians for Consumer Privacy, the group that initially spurred the legislation, does not appear to be pushing for similar nationwide regulation, and no one from the group responded to

The details of how federal regulation of data privacy and security will be supplied are unlikely to be resolved in the near term.

repeated phone calls and email messages for comment for this article.

Lan Jenson, CEO of Adaptable Security, a non-profit organization dedicated to protecting society from cybercriminals, says there needs to be a compromise between those who want IT regulation and those that believe regulation simply stifles innovation and competition. Furthermore, in the absence of specific regulations, Jenson believes there should be a concerted effort to help smaller and new market entrants address the root problem of data security and privacy. Several non-profits are working to address the issue, including:

- ▶ Jenson’s Adaptable Security, which provides pro bono or at-cost consulting services;
- ▶ the Global Cyber Alliance, which offers free cybersecurity toolkits for SMBs, and
- ▶ the National Cyber Security Alliance, which offers free informational webinars, is trying to help smaller organizations provide better data security for their users.

Regardless, coming to agreement on what should be regulated, which authority should be responsible for writing and enforcing the regulations, and how penalties should be meted out is likely to be challenging, according to David Weinberger, a senior researcher at Harvard University’s Berkman Klein Center for Internet & Society, and author of *Everyday Chaos: Technology, Complexity, and How We’re Thriving in a New World of Possibility*. “I can certainly see regulators stepping in, one way or another,” Weinberger says.

However, the details of how federal regulation of data privacy and security will be applied are unlikely to be re-

solved in the near term, largely due to multiple competing interests on both sides of the political aisle, explains Grimmelmann.

“I expect this is something we’ll still be grappling with 10 years from now,” Grimmelmann says. “If this were a simple left/right political issue, you would expect that when you have Republicans in control, they would pass legislation that they agree on, but they don’t. You have some Republicans who are highly libertarian about technology, and some who are very skeptical about tech. You have some Democrats who are very friendly with technology, and others who are very skeptical about big tech and its effect on the democracy. And as a result, you don’t get a simple coalition in Washington that’s going to say, ‘OK, you’re in power now, you’re going to impose our preferred privacy and other tech regulations.’”

Further Reading

Adaptable Security
<https://adaptablesecurity.org/>

California Consumer Privacy Act
<https://www.caprivacy.org/>

Cybersecurity Information Sharing Act
<https://www.congress.gov/bill/114th-congress/senate-bill/754>

Cybersecurity Enhancement Act of 2014
<https://www.congress.gov/bill/113th-congress/senate-bill/1353/text>

Federal Exchange Data Breach Notification Act of 2015
<https://www.congress.gov/bill/114th-congress/house-bill/555>

General Data Protection Regulation
<https://eugdpr.org/>

Global Cyber Alliance
<https://www.globalcyberalliance.org/>

National Cybersecurity Protection Advancement Act of 2015
<https://www.congress.gov/bill/114th-congress/house-bill/1731>

National Cyber Security Alliance
<https://staysafeonline.org/>

Telecommunications Act of 1996
<https://transition.fcc.gov/Reports/tcom1996.pdf>

Mark Zuckerberg: The Internet needs new rules. Let’s start in these four areas, *The Washington Post*, March 30, 2019
<https://wapo.st/2R3hYlg>

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in Lynbrook, NY, USA.

© 2019 ACM 0001-0782/19/12 \$15.00

Computing Ethics

Should Researchers Use Data from Security Breaches?

Evaluating the arguments for and against using digital data derived from security breaches.

SECURITY BREACHES AND the public exposure of internal databases, customer records, and user information are now unfortunately frequent occurrences. The data exposed by these breaches presents an ethical dilemma for computing researchers. Such datasets are potentially valuable as descriptions of actual behavior and events that are otherwise hidden from outside observers. However, researchers must confront the fact this data was obtained unethically: corporate confidentiality has been broken, and the privacy of those described in the data has been compromised.

There are a variety of contexts where laws and professional norms prohibit using unethically obtained information. In law, there are exclusion rules against using illegally obtained evidence in criminal cases. Trade secrecy law prohibits businesses from using confidential information obtained from their competitors. The 'Common Rule' in U.S. scientific research regulation and the World Medical Association's Dec-

laration of Helsinki requires human research subjects to give informed consent to participate in research.^{8,9}

Nonetheless, in some research fields there is still debate over whether researchers should use data obtained unethically by others. Medical researchers have debated whether to use the results of deadly and inhumane experimentation conducted by Nazi doctors on concentration camp prisoners and prisoners of war during the Second World War.⁶ That researchers would seriously consider using data obtained in

such grossly immoral ways shows the dilemma of using unethically obtained data is not as simple as it initially appears. Nonetheless, the arguments for and against using such data in research can guide computing researchers concerned about using data exposed by a data breach.

Unethically Obtained Data in Research

Computing researchers are not usually concerned with using data obtained by inflicting physical or psychological harm on human subjects. However, computing and Internet research may have the potential to cause harm: data analysis and computing research can be used to invade privacy and reveal information about individuals that might be used against them. Unlike medical research, the appropriate uses of big data analytics and the appropriate sources of data are not well established.⁴ In their review of published research that uses unethically obtained data, Daniel R. Thomas and his colleagues found the discussion of

There are various kinds of unethically obtained data that might interest computing researchers.

the ethical issues associated with using such data was inconsistent.⁸ While the ACM Code of Ethics lists ‘avoiding harm’ as a general principle,¹ the possible harms of using the data are not always clear.

There are various kinds of unethically obtained data that might interest computing researchers. The datasets resulting from security breaches may be password dumps, databases of internal message boards, financial and personal data, or classified information.⁷ Such information may be released onto the Internet by whistleblowers, as a result of deliberate infiltration of a secure network by outsiders, or an accidental disclosure caused by weak security practices.

We should distinguish between data unethically obtained by the researchers themselves, and data unethically obtained and released by third parties. The first case is straightforward: researchers should always reject using unethical methods. Institutional Review Boards (IRBs) and research ethics committees should be consulted if there are concerns about the methods of collecting data. Legal privacy protections also limit what researchers can collect and the methods they can employ. It is less straightforward, though, if a third party has collected data using unethical (and potentially illegal) methods and then released it publicly. Consider a whistleblower releasing confidential documents that reveal wrongdoing by governments, companies, or institutions. While it may be illegal for the whistleblower to release these documents, it is not clear researchers should ignore their contents if they are of significant research value and public interest. A blanket prohibition against using unethically obtained data may prevent socially beneficial research from occurring. The arguments for and against using unethically obtained data should therefore be considered for each individual case.

Justifying Using Exposed Data

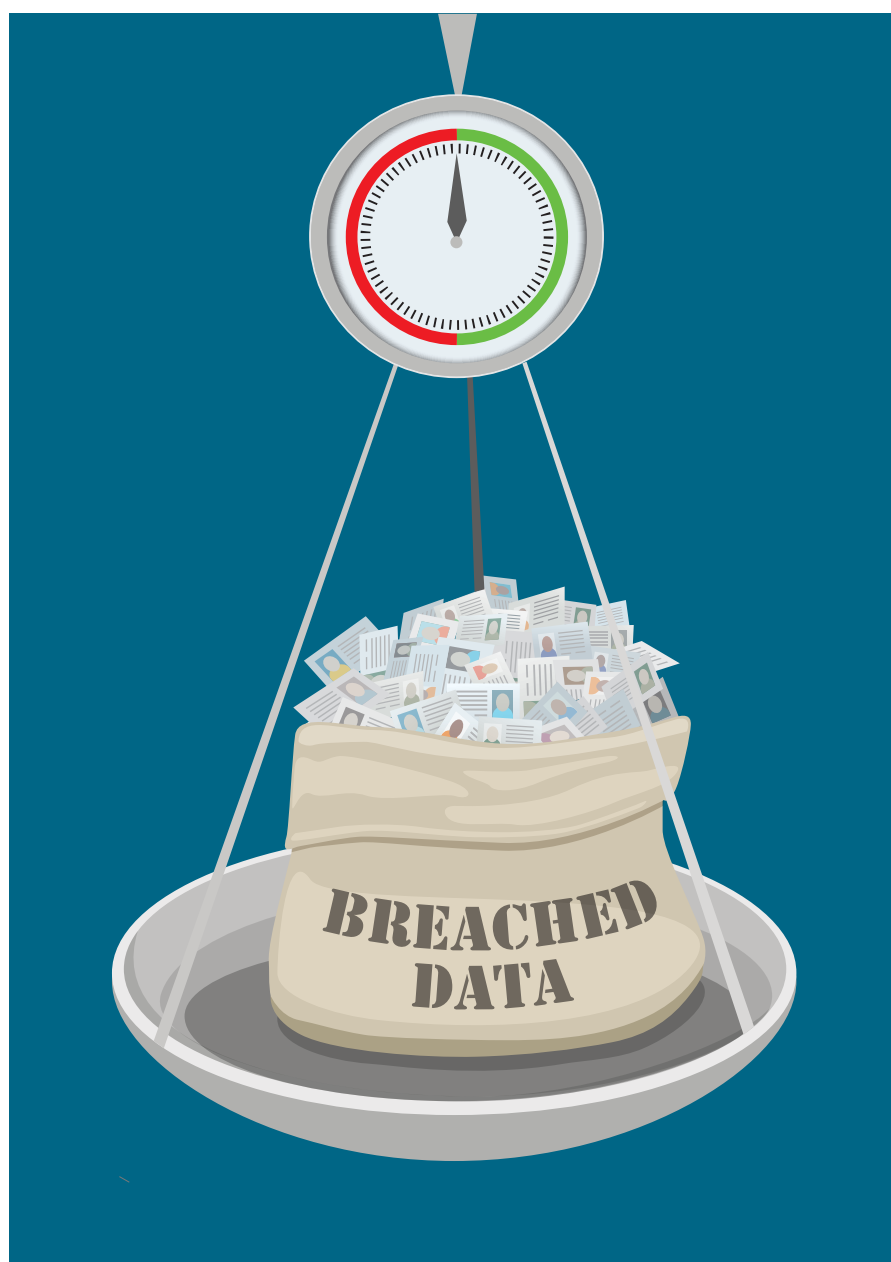
The most straightforward justification for using data exposed by a security breach is the potential benefits to society from utilizing that data outweigh the harms caused by obtaining it. The researchers must therefore offer a compelling justification for how

using the data will benefit society. If the data describes illegal or harmful activity, the obvious justification is that research using it may be used to prevent or limit such activity in the future. This recognizes the means used to obtain the data were wrong but defends the researchers’ use of it as a means to prevent or reduce another form of wrongdoing.

This argument’s effectiveness depends on both the seriousness of the unethical methods used to obtain the data and the potential significance of the research’s benefits to society. The researchers should also attempt to minimize any further harm that may occur from publishing research using

such data. For example, unethically obtained data is likely to include personal information that would otherwise have been removed or anonymized. Researchers should ensure any information that allows individuals to be identified is removed when they clean the data for analysis and publication.

Another argument is that since the data is already publicly accessible, it can be used the same way as any other publicly accessible data.⁵ The methods and motives behind the data’s release are only relevant for evaluating its quality. Given the likelihood illegal methods were used to obtain the data, the source will frequently attempt to maintain their anonymity to



avoid punishment. The source therefore is unaccountable for the data's quality. This creates the possibility that the data may have been altered or falsified for their own purpose.

This uncertainty about the data's authenticity justifies at least performing a preliminary analysis to determine whether it is genuine. Since appeals to the public benefit of using data exposed by a security breach depend on the data's accuracy, the researchers must explain how they established the data's authenticity and the likelihood it is genuine. Even if the researchers refuse to use the data in their own work, establishing whether it is likely to be genuine is useful for confirming a security breach has occurred.

However, the fact the data from a security breach is publicly accessible does not mean using it in research does not create additional risks to those it describes. For instance, publicly available information may be used to harass or threaten individuals. Jacob Metcalf rightly states that the risk to individuals from using research data depends more on the dataset's contents and the research's usage of it rather than whether the data is public, private, or de-anonymized.³ This holds for both legitimately acquired and unethically obtained data. While this might appear to downplay the significance of how the data was obtained and released, the uncertainty about data quality imposes an additional burden on using unethically obtained data. This burden is itself a potential reason to avoid using such data.

Against Using Exposed Data

The major arguments against using data exposed by a security breach are:

- ▶ the unethical methods used to obtain the data 'taints' both the data itself and any research using it as immoral;
- ▶ using such data grants the methods and those who used them an unacceptable legitimacy as 'researchers'; and
- ▶ refusing to use such data is an important statement about conducting research ethically and deters researchers from using such methods to obtain data in the future.²

The claim that unethically obtained data 'taints' research using it is both symbolic and methodologi-

cal: using the data symbolically reenacts the harm caused by obtaining it, and the unethical means of gathering it also suggests the data may be of poor quality.² For victims of security breaches, research using the exposed data may reinforce the feelings of violation and humiliation from when they discovered their data had been exposed. Methodologically, since the researchers were not involved in collecting the data, they also must confirm the data is genuine and has not been manipulated. This is part of the burden imposed by having to authenticate and clean unethically obtained data mentioned previously.

Part of the unease associated with using unethically obtained data is that it implies the researchers themselves condone the methods used to obtain it. While this is unlikely, it requires researchers to explicitly distance themselves from those who performed the security breach in publications using this data. While there may be legitimate reasons for conducting research with such data, researchers must take care to ensure the readers of their published findings are clear they do not endorse or condone the methods used to obtain the data.

An even stronger rejection of security breaches as a data source is refusing to use it in research. Such a refusal makes a clear statement about the proper methods of conducting research and obtaining data. It also deters future researchers from using such data as it means the research community will shun their work. However, adopting this position risks neglecting valuable data that may otherwise be inaccessible. If a security breach exposes data about illegal activity, this data might be useful for gaining a better understanding of how to combat it.

Handle with Caution

There are a few general conclusions to be derived from this summary of the arguments for and against using data exposed by a security breach. The risks to those described in the data and the additional burdens such data imposes on researchers means data from security breaches should only be used as a last resort. However, there are cases where obtaining data ethically is im-

possible, and compelling public interest justifications exist for analyzing it. In these cases, the burden of proof to explain the public interest in analyzing unethically obtained data is the responsibility of the researchers. IRBs and ethics committees should assist researchers in determining whether a compelling benefit to society justifies using such data.

Given the risks of using data from a security breach (both from the uncertainty of the data quality and the risks of causing further harm), researchers must ensure how they use such data minimizes these risks. The general ethical principles 1.2 ("Avoid harm") and 1.6 ("Respect privacy") of the ACM Code of Ethics apply to these cases, as they do to all computing research.¹ Published research using data from a security breach should include an ethics section where the researchers present their justifications for using the data and which IRBs and/or ethics committees reviewed and approved it.⁷ IRBs and ethics committees should also be aware of the specific concerns raised by unethically gathered data. Whether the data used is publicly accessible or not should not determine on its own whether its use poses a minimal risk to those described within it. ■

References

1. ACM Code of Ethics and Professional Conduct. Association for Computing Machinery. ACM, New York, NY, USA; 2018; <https://www.acm.org/code-of-ethics>.
2. Douglas, D.M. Should Internet researchers use ill-gotten information? *Science and Engineering Ethics* 24, 4 (Aug. 2018), 1221–1240; <https://doi.org/10.1007/s11948-017-9935-x>.
3. Metcalf, J. Big data analytics and revision of the common rule. *Commun. ACM* 59, 7 (July 2016), 31–33; <https://doi.org/10.1145/2935882>.
4. Metcalf, J. and Crawford, K. Where are human subjects in big data research? The emerging ethics divide. *Big Data and Society* 3, 1 (June 1, 2016); <https://doi.org/10.1177/2053951716650211>.
5. Poor, N. and Davidson, R. Case study: The ethics of using hacked data: Patreon's data hack and academic data standards. Council for Big Data, Ethics, and Society (Apr. 6, 2016); <http://bit.ly/2NnkscM>.
6. Rosenbaum, A.S. The use of Nazi medical experimentation data: Memorial or betrayal? *International Journal of Applied Philosophy* 4, 4 (Apr. 1989), 59–67.
7. Thomas, D.R. et al. Ethical issues in research using datasets of illicit origin. In *Proceedings of the 2017 Internet Measurement Conference*, 445–462. IMC '17. ACM, New York, NY, USA; <https://doi.org/10.1145/3131365.3131389>.
8. U.S. Department of Health and Human Services. 45 CFR 46. (July 19, 2018); <http://bit.ly/2NianOq>.
9. World Medical Association. Declaration of Helsinki—Ethical Principles for Medical Research Involving Human Subjects. (Oct. 19, 2013); <http://bit.ly/2JFEQw>.

David M. Douglas (dmdouglas256@tuta.io) is a Brisbane, Australia-based researcher in computer ethics.

Copyright held by author.



DOI:10.1145/3368095

George V. Neville-Neil

Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

Kode Vicious Koding Academies

A low-risk path to becoming a front-end plumber.

Dear KV,

I have a friend who is looking at various coding academies and asking for my advice on courses. It has been many years since I started my career as a developer, long before coding academies existed, so I am not quite sure what advice to give. What advice would you offer?

Academy Without Academe

Dear Academy,

Consider what it means to learn to be a developer in 2019. Coding academies started sometime around 2010 and were meant to address the fact that many companies were searching for people who could write the code necessary to promote their products and services, most often referred to as “front end,” because, honestly, you would never want to see what was at the “back end.” What these academies teach is neither computer science nor software engineering, but they do fill a current niche in the coding world, and they can be an entrance into the world of technology, which is fiscally rewarding. As with all things in the world of technology—including the human systems such as the schools that support that world—it is most important to look at the limitations of any product or service.

Most coding academies structure their courses such that they provide a student with a short path to a new job, often suggesting, but never promising, a job after a three- to six-month course. I can think of no better analogy than the trade schools that used to be advertised on late-night televi-



sion—the ones where you got a free toolkit after completing your certificate course in plumbing. The point of any such trade school is to provide students with the minimum skills required to practice whatever trade they are studying, and coding academies are no different. Look at the webpage for any coding academy and you will see the usual mélange of keywords you would expect for jobs building websites in 2019: HTML, CSS, JavaScript, Python, Django, Ruby on Rails, React, Angular, SQL. To those of us who spend our days working in software, only a couple of those are actually programming languages that we would recommend as part of software engineering or computer science.

When weighing any course of education, it is best to think in terms of what you get out for what you put in. The best technical educational experiences provide mental tools and frameworks for solving real-world problems across a broad spectrum, which is why a four-

year degree usually takes four years. This is not to say a four-year university program is required to learn computer science or software engineering, but it does indicate the amount of time and effort that will be required to learn skills necessary to be broadly effective in the field. Universities are expensive, and so the coding academy model should be thought of as a short-term, lower-risk way to find out if working in technology is the right fit.

Encourage your friend to pick a course that will introduce concepts that can be used into the future, rather than just a specific set of buzzword technologies that are hot this year. Most courses are based around Python. Encourage your friend to study that as a first computer language, as the concepts learned in Python can be applied in other languages and other fields. And make sure to be very direct in explaining to your friend the certificate effectively makes its holder a front-end plumber, able to unclog the series of pipes that run between businesses and consumers' wallets, and that becoming a software engineer will take quite a bit more study and practice.

KV

Related articles
on queue.acm.org

Coding Smart: People vs. Tools

Donn M. Seeley

<https://queue.acm.org/detail.cfm?id=945135>

Saddle Up, Aspiring Code Jockeys

Kode Vicious

<https://queue.acm.org/detail.cfm?id=1165762>

Programming in Franglais

Rodney Bates

<https://queue.acm.org/detail.cfm?id=1036495>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

Copyright held by author.

The Profession of IT

Uncertainty

Considering how to best navigate stability and randomness.

IN A FAMOUS episode in the “I Love Lucy” television series—“Job Switching,” better known as the chocolate factory episode—Lucy and her best-friend coworker Ethel are tasked to wrap chocolates flowing by on a conveyor belt in front of them. Each time they get better at the task, the conveyor belt speeds up. Eventually they cannot keep up and the whole scene collapses into chaos.

The threshold between order and chaos seems thin. A small perturbation—such as a slight increase in the speed of Lucy’s conveyor belt—can either do nothing or it can trigger an avalanche of disorder. The speed of events within an avalanche overwhelms us, sweeps away structures that preserve order, and robs our ability to function. Quite a number of disasters, natural or human-made, have an avalanche character—earthquakes, snow cascades, infrastructure collapse during a hurricane, or building collapse in a terror attack. Disaster-recovery planners would dearly love to predict the onset of these events so that people can safely flee and first responders can restore order with recovery resources standing in reserve.

Disruptive innovation is also a form of avalanche. Businesses hope their new products will “go viral” and sweep away competitors. Competitors want to anticipate market avalanches and sidestep them. Leaders and planners would love to predict when an avalanche might occur and how extensive it might be.

In recent years complexity theory has given us a mathematics to deal with systems where avalanches are possible. Can this theory make the needed predictions where classical statistics



cannot? Sadly, complexity theory cannot do this. The theory is very good at explaining avalanches after they have happened, but generally useless for predicting when they will occur.

Complexity Theory

In 1984, a group of scientists founded the Santa Fe Institute to see if they could apply their knowledge of physics and mathematics to give a theory of chaotic behavior that would enable professionals and managers to move productively amid uncertainty. Over the years the best mathematical minds developed a beautiful, rich theory of complex systems.

Traditional probability theory provides mathematical tools for dealing with uncertainty. It assumes the uncertainty arises from random variables that

have probability distributions over their possible values. It typically predicts the future values of the variable by computing a mean of the distribution and a confidence interval based on its standard deviation. For example, in 1962 Everett Rogers studied the adoption times of the members of a community in response to a proposed innovation.⁵ He found they follow a Normal (Bell) curve that has a mean and a standard deviation. A prediction of adoption time is the mean time bracketed by a confidence interval: for example, 68% of the adoption times are within one standard deviation of the mean and 95% are within two standard deviations.

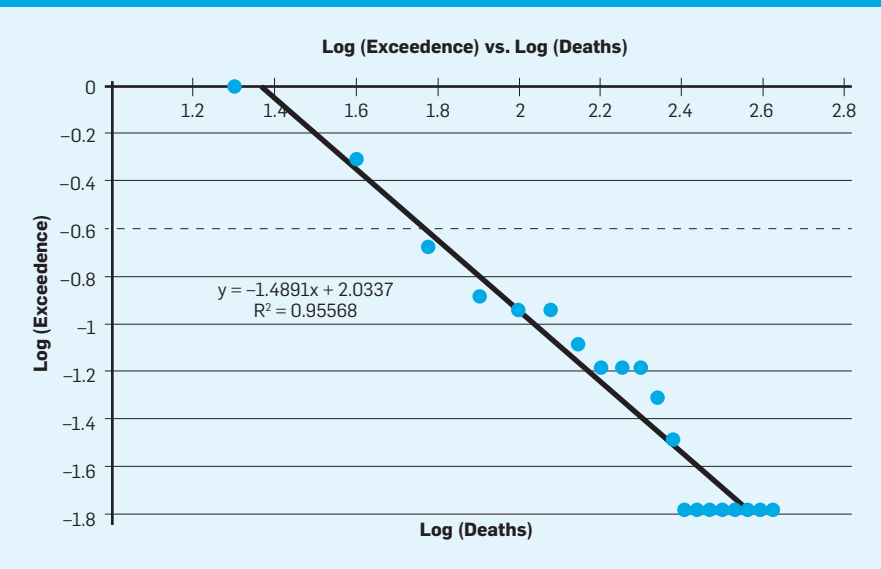
In 1987, researchers Per Bak, Chao Tang, and Kurt Wiesenfeld published the results of a simple experiment that

demonstrated the essence of complexity theory.⁴ They observed a sand pile as it formed by dropping grains of sand on a flat surface. Most of the time, each new grain would settle into a stable position on the growing cone of sand. But at unpredictable moments a grain would set off an avalanche of unpredictable size that cascaded down the side of the sand pile. The researchers measured the time intervals between avalanche starts and the sizes of avalanches. To their surprise, these two random variables did not fit any classical probability distribution such as the Normal or Poisson distributions. Instead, their distributions followed a “power law,” meaning the probability of a sample of length x is proportional to x^{-k} , where k a fixed parameter of the random process. Power law distributions have a finite mean only if $k > 2$ and variance only if $k > 3$. This means a power law with $k < 2$ has no mean or variance. Its future is unpredictable. When $2 < k < 3$, the mean is finite but not the confidence interval. Bak et al. had discovered something different—a random process whose future could not be predicted with any confidence.

This was not an isolated finding. Most of the random processes tied to chaotic situations obey a power law with $k < 3$. For example, the appearance of new connections among Web pages is chaotic. The number of Web pages with x connections to other pages is proportional to $1/x^2$ —the random process of accumulating links produces $1/4$ as many pages with $2x$ connections as with x connections. This was taken as both bad and good news for the Internet. The bad news is that because there are a very few “hubs”—servers hosting a very large number of connections—an attacker could shatter the network into isolated pieces by bringing down the hubs. The good news is the vast majority of servers host few connections and thus random server failures are unlikely to shatter the network. What makes this happen is “preferential attachment”—when a new Web page joins the network, it tends to connect with the most highly connected nodes already in the network. Startup company founders try to plot strategies to bring about rapid adoption of their technologies and transform their new services into hubs.

Hundreds of processes in science and engineering follow power laws and their key variables are unpredictable. Innova-

Log-log plot of the exceedance versus intervals between terror attacks follows a straight line. Exceedance is the probability that an interval is greater than x (a tail of the distribution). A straight line on log-log plot is the signature of a power law; here the slope is -1.4 , telling us the tails of the distribution are a power law $y=x^{-1.4}$. Because 1.4 is less than 2, this distribution has no finite mean or standard deviation: the time to next terror attack is unpredictable.



tion experts believe innovations follow a power law—the number of innovations adopted by communities of size x is proportional to x^{-2} —not good news for startup companies hoping to predict their innovations will take over the market.

Later Bak¹ developed a theory of unpredictability that has subsequently been copied by popular writers like Nassim Nicholas Taleb and others.⁶ Bak called it punctuated equilibrium, a concept first proposed by Stephen Jay Gould and Niles Eldredge in 1972.³ The idea is that new members can join a complex system by fitting in to the existing structure; but occasionally, the structure passes a critical point and collapses and the process starts over. The community order that has worked for a long time can become brittle. Avalanche is an apt term for the

moment of collapse. In the sand pile, for example, most new grains lodge firmly into a place on the pile but occasionally one sets off an avalanche that changes the structure. In the Internet, malware can quickly travel via a hub to many nodes and cause a large-scale avalanche of disruption. In an economy, a new technology can suddenly trigger an avalanche that sweeps away an old structure of jobs and professions and establishes a new order, leaving many people stranded. Complexity theory tells us we frequently encounter systems that transition between stability and randomness.

Punctuated equilibrium appears differently in different systems because self-organization manifests in different ways. In the Internet, it may be the vulnerability to the failure of highly connected hubs. In a national highway system, it may be the collapse of maintenance as more roads are added, bringing new traffic that deteriorates old roads faster. In geology, it may be the sudden earthquake that shatters a stable fault and produces a cascade of aftershocks. In a social system, it may be the outbreak of protests when people get “fed up.”

Explanations but Not Predictions

What can we learn from all this? Many systems have a strong social component, which leads to forms of preferential attachment and power-laws governing the degrees of connectivity in the

Hundreds of processes in science and engineering follow power laws and their key variables are unpredictable.

social network. These systems are susceptible to sudden changes of structure of unpredictable onset and extent. The best we can say is the conditions for avalanche are present but we cannot say with any certainty the avalanche will actually happen or, if it does, what its extent will be. In other words, we are able to explain an avalanche after it happens but we are profoundly unable to predict anything about it before it happens.

Earthquake preparedness is an example in nature that does not depend on humans. Seismic experts can tell us where the fault lines are and compute the probabilities of earthquake on different faults. They cannot, however, predict when an earthquake will happen or how large it will be. In effect they are trying to predict when an earthy avalanche—collapse of structure in a section of earth's crust—will happen. Similarly, snow experts know when conditions are “ripe” for an avalanche and can call for evacuating the area. But they cannot know exactly where a snow avalanche may start, or when, or how much snow will sweep down. These experts call on people to be prepared but few actually heed the advice and lay in necessary supplies or make necessary contingency plans.

Navigating in Uncertainty

Complexity researchers have turned to simulations of complex systems to see when avalanches happen and how large they are. These simulations often reveal regularities in the state spaces of those systems that can be usefully exploited to make predictions.

What are more pragmatic things we can do to cope with uncertainty? We can learn some lessons from those who must deal with disasters such as fires, earthquakes, floods, or terror attacks. Their data shows the times between events and sizes of events follow power laws and cannot be predicted. Their coping strategy boils down to preparedness and resiliency. Preparedness means to have recovery resources standing by in case of need. Resiliency means to rapidly bounce back and restore order and function.

They have worked out strategies to identify the situations most “ripe” for an avalanche. For instance, the power law for terror attacks shows that attacks tend to cluster in time at a given

One of the most difficult environments to navigate is the social space in which we perform our work. This space is dominated by choices that other people make beyond our control.

location. Thus, a next attack is more likely at the same location as the current attack. The preparedness strategies include rapid mobilization of law enforcement just after an attack to counter the tendency for a new attack, and to identify optimal geographic locations for positioning recovery resources and supplies. Resilience strategies include rapidly mobilizing technicians and artisans to restore broken communications and facilities.

Uncertainty in Professional Work

What can we do when we find ourselves in chaotic situations and must still navigate through the uncertainty to achieve our goals?

One of the most difficult environments to navigate is the social space in which we perform our work. This space is dominated by choices that other people make beyond our control. When we propose innovations, we are likely to encounter resistance from some sectors of our community that do not want the innovation; they can be quite inventive in finding ways to block our proposals.² When we start new projects or even companies, we do not know whether our plans are going to take off or just wither away. Even in normal everyday working environments, conflicts and contingencies suddenly arise and we must resolve them to keep moving forward.

The analogy of a surfer is useful in approaching these situations. A surfer aims to ride the waves to the shore without losing balance and being swept under. The waves can be turbulent and unpredict-

able. The surfer must maintain balance, ride the crests moving toward the shore, and dodge side waves and cross currents. The surfer may need to jump to a new wave when the time is right, or quickly tack to avoid an unfavorable current or wind. Thus, the surfer generates a path through the turbulent waves.

In the social space, waves manifest as groups of people disposed to move in certain directions and not in others—sometimes the waves appear as fads or “memes” and they have a momentum that is difficult to divert. As a professional, we become aware of these waves and try to harness them to carry us toward our goal. As each surprise pops up, we instinctively look for openings into which we can move—and, more importantly, we create openings by starting conversations that assuage the concerns of those whose resistance threatens to block us. These little deals cut a path through the potential resistance and get us to our goal.

The lesson here is that we listen for the waves, ride their momentum toward our goal, and make adjustments by creating openings in our conversations with other people. At its best, the complexity theory helps us understand when a process is susceptible to unpredictable avalanches. We move beyond the limitations of the theory by generating openings in our conversations with other people. **C**

References

1. Bak, P. *How Nature Works: The Science of Self-Organized Criticality*. Springer-Verlag, 1996.
2. Denning, P. Winning at innovation. *IEEE Computer* (Oct. 2018), 32–39.
3. Eldredge, N. and Gould, S.J. Punctuated equilibria: An alternative to phyletic gradualism. In T.J.M. Schopf, Ed., *Models in Paleobiology*. Freeman Cooper, San Francisco, CA, 82–115.
4. Lewis, T.G. *Bak's Sand Pile: Strategies for a Catastrophic World*. Agile Press (2011), 382.
5. Rogers, E. *Diffusion of Innovations* (5th edition). Free Press, 2003.
6. Taleb, N.N. *The Black Swan: The Impact of the Highly Improbable*. Random House, 2007, 2010.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, USA, is Editor of *ACM Ubiquity*, and is a past president of ACM. The author's views expressed here are not necessarily those of his employer or the USA federal government.

Ted G. Lewis (tedglewis@redshift.com) is an author and consultant with more than 30 books on computing and hi-tech business, a retired professor of computer science, most recently at the Naval Postgraduate School, Monterey, CA, USA, a Fortune 500 executive, and the co-founder of the Center for Homeland Defense and Security at the Naval Postgraduate School, Monterey, CA, USA.

Copyright held by authors.

Viewpoint

Public Entrepreneurship and Policy Engineering

Training the next generation of leader and problem solver.

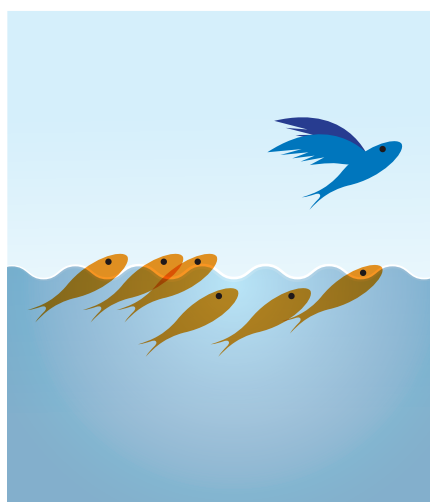
SCIENCE AND TECHNOLOGY have progressed exponentially, making it possible for humans to live longer, healthier, more creative lives.

The explosion of Internet and mobile phone technologies have increased trade, literacy, and mobility.¹¹ At the same time, life expectancy for the poor has not increased and is declining.⁹

As science fiction writer William Gibson famously quipped, the future is here, but unevenly distributed. With urgent problems from inequality to climate change, we must train more passionate and innovative people—what I call public entrepreneurs—to learn how to leverage new technology to tackle public problems. Public problems are those compelling and important challenges where neither the problem is well understood nor the solution agreed upon, yet we must devise and implement approaches, often from different disciplines, in an effort to improve people's lives.

Problem solving has been identified as one of the most important skills a graduate of engineering and computer science, as well as graduates of other professional schools, must have in the 21st century. Yet, as someone who has been tenured in both law and engineering and taught in a public policy school, I can report that our universities are failing to teach future professionals how to tackle complex problems in a rapidly changing world.

The primacy of technology in our



daily lives combined with the urgent need to design and implement solutions to public problems require a new curriculum of public entrepreneurship.

Public entrepreneurship teaches students to tackle public problems in the public interest. First, it demands working together in teams and with real-world partners across disciplinary silos. Second, it teaches participants to go beyond vague issues to define actionable problems. Third, public entrepreneurs learn to use the tools of both data and collective intelligence to get smarter about both problems and solutions. Fourth, they learn to design solutions together with those they are trying to help by adopting more participatory and democratic ways of working and, finally, they learn how to implement measurable solutions in the real world that improve people's lives.

What's Wrong?

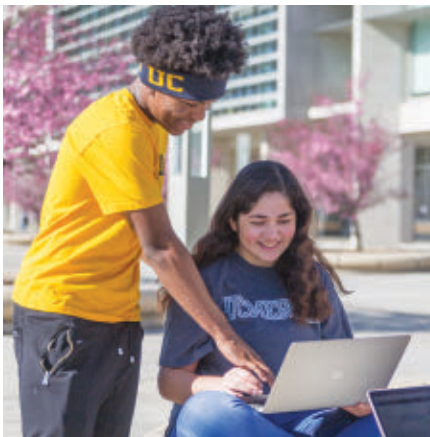
Done right, engineering and computer science teach technological craft, but all too often without complementary instruction in how to implement solutions in the real world of *institutions*. In fact, engineering is too often taught without regard for cultural, social, and political context.³ Internships, while useful, are no substitute for acquiring the formal skills of problem solving.

For law students, too, we give them too limited a toolkit. Whereas public interest litigation, strategic use of contracts, and knowledge of how to craft legislation and regulation are vital mechanisms for public problem solving, the agile and flexible tools of technology, data, and innovation have been woefully absent from the curriculum.

Our public policy schools are similarly out of date. For example, of the top 25 public policy schools as ranked by *U.S. News and World Report* in 2019, none required students^a to take even basic coursework in data science. Former Woodrow Wilson School Dean and head of New America Foundation Anne-Marie Slaughter^b writes that our public policy schools are teaching a method of problem solving that is “built in a different time and for a different time—an era with fewer citizens, a slower pace of information dissemination, and a data capacity that is a fraction of what we have today.”⁸ In a recent

a <http://www.naspaa.org>

b http://en.wikipedia.org/wiki/anne-marie_slaughter



COMPUTER SCIENCE @

UC MERCED



Ranked among the best public universities in the nation by U.S. News and World Report

PhD, MS, and undergraduate programs in Computer Science

Situated near Silicon Valley

Easy access to California's National Parks

LEARN MORE

<https://engineering.ucmerced.edu>

<https://graduatedivision.ucmerced.edu>

<https://eecs.ucmerced.edu>

UNIVERSITY OF CALIFORNIA
MERCED

public statement^c by a group of leading deans and educators convened by Stanford University, we wrote: “In the United States, and other consolidated democracies, the system of educating and training people to solve public problems is radically insufficient. Often such education and training, especially for professionals, simply does not exist.”

What's the Solution?

Back in 2015, 120 U.S. engineering and computer science school deans announced plans to educate a new generation of engineers expressly prepared to tackle some of the most pressing issues of the 21st century. Moreover, computer science and engineering students as much as their law and policy counterparts are keen to “do stuff that matters.”

Create transdisciplinary teams. First, schools must create real-world problem-based learning opportunities that cut across the natural and social sciences by creating teams of engineering and computer science, law, and policy students to work on public interest problems in collaboration with outside institutions. Between the National Academy of Engineering's (NAE) Fourteen Grand Challenges^d and the United Nations 17 Sustainable Development Goals (SDGs)^e there is no shortage of important public problems. At NYU Tandon's Ability Lab,^f students across CS, engineering, design, and occupational therapy are designing better tools to improve the lives of those with disabilities.

Rensselaer Center for Open Source,^g the centerpiece of its computer science student-run programs endeavors to “empower students to develop open source solutions to real-world problems.” In one recent project, a dozen students are helping community organizations to develop simple free tools to advance their societal impact.

Although there are often collaborative programs across different branches of engineering, such as computer

and biological engineering, however, most professional schools teach problem solving only with the toolkit of their own discipline as a result of a century-long effort to professionalize a unique science for each field.¹⁰ Now we are seeing the consequences of such siloes with students underprepared to work in global cross-disciplinary and diverse teams.

Over the years of teaching public problem solving in law, policy, and engineering schools, I have found that exposing students to one another's complementary substantive knowledge and diverse problem-solving heuristics and vocabularies enables teams to get more done.

Learn to define an actionable problem. Second, professional schools must teach problem solving beginning with actionable problem definition,^h rather than by handing students a problem. Too often students are taught to solve well-structured problems working from preexisting cases.

Even so-called capstone or research assistantships assign prescribed tasks to students. But real-world problems are not well structured and context specific. Empirical research has shown that only learning to solve well-structured problems does not readily enable graduates to tackle open-ended, complex real-world problems.⁷

A real innovator must discover the problem rather than work on a problem already presented.⁵ This is what educators refer to as problem-based rather than problem-solving or even project-based learning and requires learning the epistemic craft of how to define a problem, its historical and social context, and root causes.¹¹ In *How We Think*, John Dewey claimed this process of defining the problem reflects the essence of the development of complex thought and that is why Maastricht University uses a problem-based learning model for all its teaching.^{i,j,2,4}

Use diverse tools. Third, it is a commonplace to say that defining a problem depends upon understanding root causes. But, to do so well, teams must

c <https://fsi.stanford.edu/publicproblemsolving/docs/statement-education-public-problem-solving>

d <https://www.wired.com/2008/02/and-the-14-big/>

e <https://sustainabledevelopment.un.org/sdgs>

f <http://ability.nyu.edu>

g <http://rcos.rpi.edu>

h <https://innovation.nj.gov/skills/modules/problem-definition.htm>

i <http://www.maastrichtuniversity.nl/education/why-um/problem-based-learning>

j <http://openseventeen.org>

use both quantitative and qualitative methods—getting smarter from both data and human insight to frame the problem to solve.

This means abandoning a dogmatic adherence either to human-centered design or data analysis as the exclusive means of problem discovery. The popularity of design thinking as evidenced by an increasing number of design science courses and programs, on the one hand, and of data science pedagogy, on the other, have led to a headlong rush to embrace one or the other set of tools.

Whereas data might reveal where gun violence is occurring, only talking to those with relevant professional know-how as well as police, victims, and families will reveal why it is happening. Students across disciplines must develop both sets of skills. In the Open Seventeen program, a partnership among Tsinghua, NYU, and the Universities of Zurich and Geneva, global students from computer science and informatics, design, history, policy and more, received online project coaching from the Governance Lab and learn to apply both human-centered design and data analytical methods to advance projects that respond to one of the 17 SDGs.^k

From design to implementation. Fourth, we need to teach students the undervalued and more challenging task of implementing solutions, not just designing them, by assessing feasibility in the context of real-world institutions. Even public policy schools, where one would assume this is taught, says Frank Fukuyama^l “train students to become capable policy analysts, but with no understanding of how to implement those policies in the real world.”

Tackling social problems requires a deep understanding for engineers and computer scientists, too, of how to work with institutions to take advantage of their power, reach, and resources. Thus, to develop their lifelong clinical problem-solving capabilities, the would-be public entrepreneur must learn a deep understanding of how bureaucracy and politics can be harnessed for impact. This is why at Purdue the Engineering Projects in

Community Service (EPICS) project^m offers students the opportunity to learn about the social context of the projects they do. More than clever ideas and ingenious gadgets, our ability to solve problems depends upon people with the willingness and wherewithal to deliver and spread impact.

From social enterprise to public entrepreneurship. To be sure, most schools today offer some kind of program in *private* entrepreneurship. Yet the pedagogy of problem solving taught in those clubs and classes has critical limitations.⁶ Private entrepreneurship focuses on the ego of the individual and the ability of a person to devise their own original solution, launch an app, or start a company.

But true public problem-solving demands solutions that are legitimate as well as effective. This means that, rather than learning to develop solutions oneself, students must learn participatory and democratic methods for defining problems and developing solutions *with* rather than *for* communities.

Take the example of Indian public entrepreneur and biophysicist Samir Brahmchari, former director general of the Council of Scientific and Industrial Research of the Government of India. In India, thousands of primarily poor people die every year from tuberculosis. Yet there has been no new TB treatment developed in 40 years and resistance to existing drugs is increasing. So Brahmchari became the chief mentor for the Open Source Drug Discovery project,ⁿ a crowdsourcing effort to “provide affordable healthcare to the developing world by providing a global platform where the best minds can collaborate and collectively endeavor to solve the complex problems associated with discovering novel therapies for neglected tropical diseases.” By recruiting college students, academics, and scientists from around the world and across India, most in remote villages not elite universities, Brahmchari created his own inexpensive army to collect, annotate, and extract information from the scientific literature on the TB pathogen. With a \$12 million grant, he coordinated the incremental contributions of 7,500 participants from 130

countries and began clinical trials for a new experimental drug at 20% of the cost of a traditional drug in 2014.¹

Brahmchari is paradigmatic of the new public entrepreneur who works differently, taking advantage of new technologies, especially the tools of big data and collective intelligence—but also the convening power of institutions—to take on difficult and seemingly intractable public problems.

Complementing traditional university education in computer science and engineering as well as law and policy with public problem solving in multidisciplinary teams to design and implement workable solutions with public institutions would allow us to cultivate more such public entrepreneurs, transforming how we educate while producing the leaders and problem solvers committed to improving people’s lives that we so desperately need. 

References

1. Ardal, C. et al. Open Source Drug Discovery in Practice: A Case Study National Center for Biotechnology Information (Sep. 20, 2012); <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3447952/>
2. Boydston, J. *John Dewey: The Later Works 1925–1953, Vol. 8 1933 Essays and How We Think, Revised Edition*. Southern Illinois University Press, Carbondale, 1986.
3. Cech, E. and Sherick, H. Depoliticization and the structure of engineering education. In *International Perspectives on Engineering Education* (2015).
4. Farra, H. The reflective thought process: John Dewey re-visited. *The Journal of Creative Behavior* (Mar. 1988); <https://doi-org.proxy.library.nyu.edu/10.1002/j.2162-6057.1988.tb01338.x>
5. Getzels, J. and Csikszentmihalyi, M. *The Creative Vision: A Longitudinal Study of Problem Finding in Art*. Wiley, New York, 1976.
6. Giridharadas, A. *Winners Take All: The Elite Charade of Changing the World*. Knopf Doubleday Publishing Group, New York, 2018.
7. Jonassen, D. *Learning to Solve Problems: An Instructional Design Guide*. Wiley, New York, 2003.
8. McGuinness, T. and Slaughter, A. The new practice of public problem solving. *Stanford Social Innovation Review* (2019); https://ssir.org/articles/entry/the_new_practice_of_public_problem_solving#
9. National Academies of Sciences, Engineering, and Medicine. *The Growing Gap in Life Expectancy by Income: Implications for Federal Programs and Policy Responses*. The National Academies Press, Washington DC, 2015; <https://doi.org/10.17226/19015>
10. Noveck, B. *Smart Citizens, Smarter State: The Technologies of Expertise and the Future of Governing*. Harvard University Press, Cambridge, 2015.
11. Savin-Baden, M. The problem-based learning landscape. *Planet* (2001); <https://doi.org/10.11120/plan.2001.00040004>

Beth Simone Noveck (noveck@thegovlab.org) is a professor in the Technology, Culture, and Society department at New York University’s Tandon School of Engineering and director of The Governance Lab, Brooklyn, NY, USA. Her new book *Public Entrepreneurship* will appear with Yale University Press in 2020; see <https://www.publicentrepreneur.org/>.

Copyright held by author.

k <http://openseventeen.org>

l https://en.wikipedia.org/wiki/francis_fukuyama

m <http://www.ijee.ie/articles/vol21-1/ijee.1549.pdf>

n <http://www.osdd.net/>

Investigating the emerging black market of retail email account hacking services.

BY ARIANA MIRIAN

Hack for Hire

A SINGLE EMAIL address often underpins one's entire online identity, from banks, to business, to social media profiles and more. This identity is used not only when registering for this multitude of services, but also when passwords for these services must be reset. Thus, an attacker gaining access to an email account poses the risk of compromising all the other services tied to that account as well. Politicians, journalists, and cryptocurrency folks have all been the victims of targeted attacks that started with access to their email accounts that then wreaked havoc on other online accounts tied to those email accounts.^{3,7,9}

Since email accounts can provide a wealth of information, many types of attacks target them: password guessing, access token theft, password reset fraud, and phishing, to name a few. Email providers have added mechanisms such as security questions, spam filtering, and two-factor authentication to limit the success rate

of these attacks.^{4-6,12} These defenses prevent many compromises, thus increasing the sophistication and time needed to access accounts. Targeted attackers, however, are willing to put in the extra effort needed to access an account in the face of these large-scale defenses.

While targeted attacks are often thought of as requiring nation-state capabilities, there is an emerging black market for “hack-for-hire” services, which provide targeted attacks to anyone willing to pay a modest fee. These services purport to be able to break into the accounts of a variety of different email providers, an example of which is shown in Figure 1. As these services are just emerging, little is known about how they attack their victims and how much of a risk they pose.

To understand this risk, we investigated the hack-for-hire black market, identifying 27 retail email account hacking services and purchasing these services from them. Using covert identities, we engaged with these services to break into purported “victims”—in truth, Google accounts that we controlled. Working with Google, we recorded both our interactions with the hijackers and how these hijackers tried to attack our victims.

To Catch a Hijacker

As a whole, the targeted hijacking black market was riddled with scams, but a handful of services launched sophisticated attacks that leveraged phishing as their main attack vector. These attacks were persistent, personalized, and able to bypass SMS two-factor authentication (2FA). Using signals derived from these attacks to identify other victims, we estimate that attackers target about one in every million Google accounts. Given the sophistication of the phishing attacks, we believe that the best line of defense for at-risk users is to protect accounts with universal 2nd factor (U2F) security keys as a 2FA mechanism. U2F security keys protect against sophisticated phishing attacks because the U2F protocol validates the domain before



sending the 2FA code, preventing a user from getting phished. Though the focus of this investigation was on Google accounts, the lessons learned generalize well across email providers.

Discovery of services. The investigation of hack-for-hire services began by searching English, Chinese, and Russian black market forums for advertisements related to targeted account hijacking. We also searched Google for hijacking-specific keywords to identify services with public-facing storefronts, and we contacted the abuse teams of large Internet companies for leads on any such services they were tracking. In all, 27 prospective hack-for-hire services were identified. The majority of these services advertised in Russian, and ranged anywhere in price from \$23 to \$500 per contract.

Posing as a buyer. For every hack-for-hire service contacted, we communicated via a unique “buyer persona” to pro-

tect our identity and to avoid linking our interactions across services. Each persona involved selecting a name in the native language of the hack-for-hire service. For example, if the service advertised in Russian, then we chose a common first and last name in Russian for our persona. We also created a Google account for each persona to use for all email communication. For non-English services, a native speaker performed all translation when communicating.

Selecting a victim. When contracting hack-for-hire services, we created a victim persona to serve as a target. The victim persona was given a large digital footprint to craft a realistic online presence. This meant creating a name and Google account for the victim in a similar fashion to the buyer persona. The victim persona’s inbox was populated with a subset of messages from the Enron email corpus to give the impression that the Google account was in ac-

tive use.² We replaced the names and other identifying information from the Enron messages with the victim’s information. Moreover, the victim persona’s Gmail address was protected by SMS 2FA, the most widely used form of 2FA today.¹ This was used to determine if the hack-for-hire services would be able to bypass this type of protection.

In addition, we created a Web page that advertised a small business that the victim either owned or worked at. We purchased the domains of the Web pages from auction to ensure each domain had prior history. We also purchased privacy protection for each of the domains to protect the registration information (one recent study showed that 20 percent of domains are protected in this fashion, so we did not expect privacy protection to raise any red flags⁸). This webpage linked to the victim’s email address, as well as a fictitious associate’s email address. In this way we could determine if

the hack-for-hire services would attack the associate as a way to gain access to the victim. We also created a Facebook page for the victim to see if the hack-for-hire services would use it in their attacks. All items on the Facebook page were private (a public user would not be able to see these items) except the About Me section, where the victim's Web page was listed (as a personal advertisement for the victim's business).

Monitoring attacks. Because of uncertainty over which attack methods these hack-for-hire services would use, we created an extensive monitoring infrastructure that would inform us when an unauthorized user entered and modified the Google account. We also monitored the websites to record all visitors. This monitoring infrastructure contained: a Google app script for Gmail in every account; Google logs; and a network capture of all traffic to the fictitious websites.

The Google app script loaded into Gmail for each account was a modifica-

tion from one used in a previous study.¹¹ The Google app script would send information to a server controlled via a proxy indicating whether the script was still connected to the account and whether there were any changes to the account. For example, the Google app script would indicate whether a new message had appeared in the inbox or spam folders, if a message was moved to trash, or if any messages marked as unread were read by a user. This logging recorded the actions of the attackers once they were in the account.

We also were able to analyze any login activity to our victim personas' Google accounts. These logs, captured and analyzed by our Google colleagues, recorded login attempts into the account and their origins, brute-force attempts, and whether 2FA was triggered on the account for a suspicious login attempt.

Finally, we captured all network traffic to each victim persona's website.

As mentioned previously, the website address was linked on the About Me section of the victim's Facebook page. If an attacker was able to find the website via the Facebook page, this would be apparent in the network capture. The network capture would also show if any of the hacking attempts to the Google accounts were from the same IP addresses that were visiting the Web pages.

Legal and ethical considerations. Since this study involved engaging with actors that were performing illegal activities, there were legal and ethical questions to consider.

Legally, there were two concerns: unauthorized access into Google accounts and violating Google's terms of service. In the U.S., as in many other countries, unauthorized access into an electronic account is illegal. Hiring services to perform this act could be considered aiding and abetting; however, since the email accounts were directly under our control and we were acting in collaboration with the service provider (Google), we were explicitly authorizing entities to access our accounts. Moreover, creating fake Google accounts violates Google's terms of service, but this study was approved by both Google and the general counsel for UC San Diego, before the work was started.

Although this study is not considered human rights research by our institutional review board because we were measuring organizational behaviors and not behaviors of an individual, there were other ethical considerations. By creating fictitious victim and buyer personas, we removed the possibility of any individual being harmed during this study. Moreover, we interacted with these services within the scope of their terms and paid them if they were successful. We believe the lessons learned from this study outweigh the cost of supporting these services by paying them.

Hack-For-Hire Playbook

The controlled experiment and logging infrastructure allowed examination of the playbook that attackers use to take over a victim account. Only five of the 27 services we contacted actually attempted to break into the victim's Google account. Note that the "success" of a hack-for-hire service was dependent on our actions: in some cases,

Figure 1. Example of a hack-for-hire advertisement.

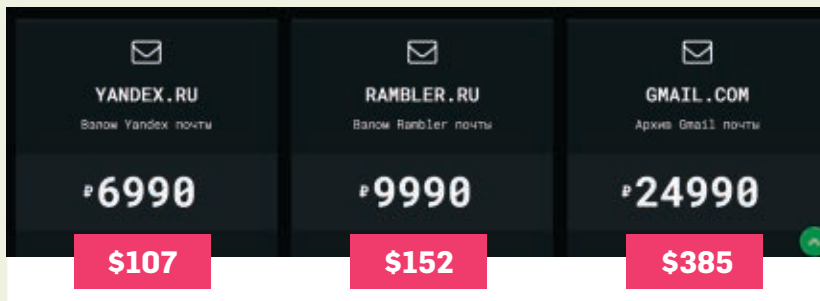
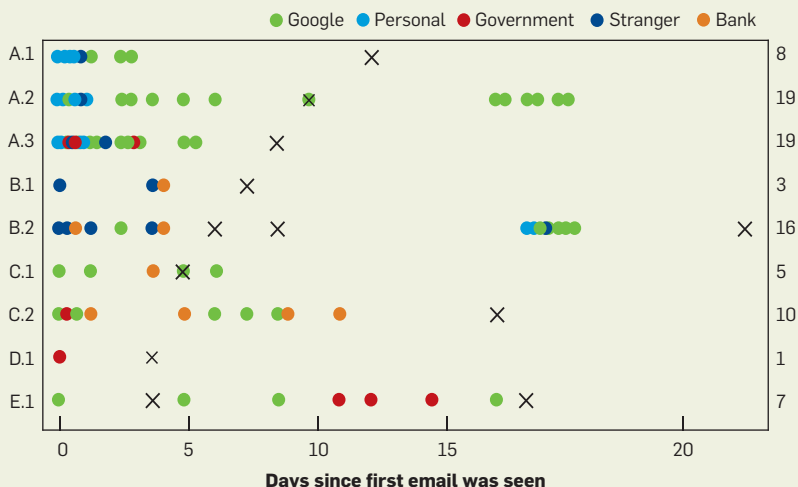


Figure 2. Lures used to try to access a victim account.



we provided the password or 2FA code when prompted by a phishing page, while in other cases we did not in order to see how the service would adapt.

Overall, we never observed any brute-force login attempts, communication with the victim's Facebook account, or communication with the associate's email. Of the five services that attempted to gain access to the account, one sent the victim malware executable via an email message. While we were not able to run the executable in a virtual machine sandbox, VirusTotal reported the malware executable was a remote access trojan. The other four services used phishing as their primary attack vector. I share our key findings here, but more details can be found in the full article.¹⁰

Email lures and phishing. All of the attacks started with an email lure to the victim account. These lures impersonated some trusted or authority figure, presumably to spur the victim into clicking on the link. There were five types of lures across all phishing messages: those impersonating an associate persona, a bank, a stranger, a government entity, or Google (as illustrated in Figure 2). The associate lures leveraged trust to get the user to click on an “image” (which led to a phishing page), while the stranger lures consisted of an unknown person emailing the user with an “image” or link. However, the government, bank, and Google lures conveyed a sense of urgency in their messages. Figure 3 shows examples of a government lure (translated into English) and a Google lure.

On average, attackers sent 10 messages over the course of 25 days and used different lures during their persistent attacks. Figure 2 illustrates this behavior, showing the time since the attackers sent their first email message, the type of lure for each message, and when we clicked on the lure (potentially halting any future attempts). An X indicates when we clicked on a link in a message sent to a victim. Each row corresponds to one victim and numbers on the right denote the total number of emails sent by a service. The most popular lure mimicked Google, followed by associates, then lures from strangers.

Of the services that sent personalized lures, all but one asked the buyer persona for more details ahead of time (such as the name and email address of

a known associate). One service was able to construct personalized lures without additional information from the buyer persona, indicating that this service searched and found the online website constructed for the victim. We also purchased two other contracts from this service, however, and in both cases the operator asked the buyer persona for details on the victim. These differences in behavior suggest these services have multiple people working behind them.

As mentioned previously, all but one of the services relied on phishing as their main attack vector. Clicking on the phishing link took us to a landing page that looked like the Google login page. After entering our password, we were taken to a page that prompted us for our 2FA code. All of the services that were able to access the account did so by phishing the SMS 2FA code, bypassing this security protection.

Live adaption. While most of the services accounted for the 2FA code in their initial phishing flow, two exhibited phishing attacks that adapted to

obstacles. These two services, in their initial phishing flow, did not account for a 2FA code. Their phishing flow collected the password and then tried to log into the Google account, which was successfully blocked by 2FA. After realizing this, both services sent the victim additional email messages with a different structure from the ones previously sent. One of these two services sent a new message that, when clicked, would request both the password and the 2FA code. The other service also changed its flow to account for the 2FA code, and in doing so sent phishing messages that looked similar to those sent from a third service. These similarities suggest the use of common tools among services.

Note that these services were able to bypass 2FA because they phished the SMS 2FA code from the user. As noted earlier, SMS 2FA is the most widely used form of 2FA, and we wanted to see if those who use it would be susceptible to a hypothetical hack-for-hire attack. To prevent hack-for-hire services from gaining access to an ac-

Figure 3. Examples of a government lure and a Google lure.

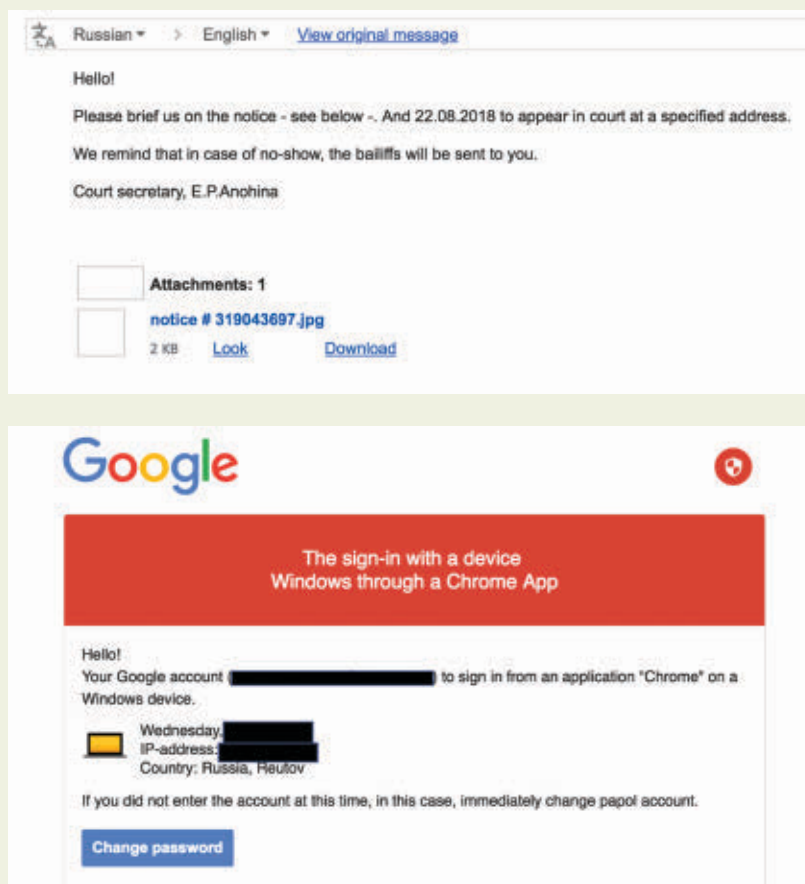


Figure 4. Purported prices to access various accounts.

Target	Service A	Service B	Service C	Service D*	Service E*
Mail.ru	\$77	\$77	\$62	\$54	\$77
Rambler	\$152	\$108	\$77	\$77	\$108
Yandex	\$106	\$108	\$77	\$77	\$108
Gmail	\$384	\$385	\$92	\$77	Negotiable
Yahoo	\$384	\$231	\$92	—	—
Facebook	\$306	—	—	—	—
Instagram	\$306	—	—	—	\$231

Figure 5. Monthly prices for service A.

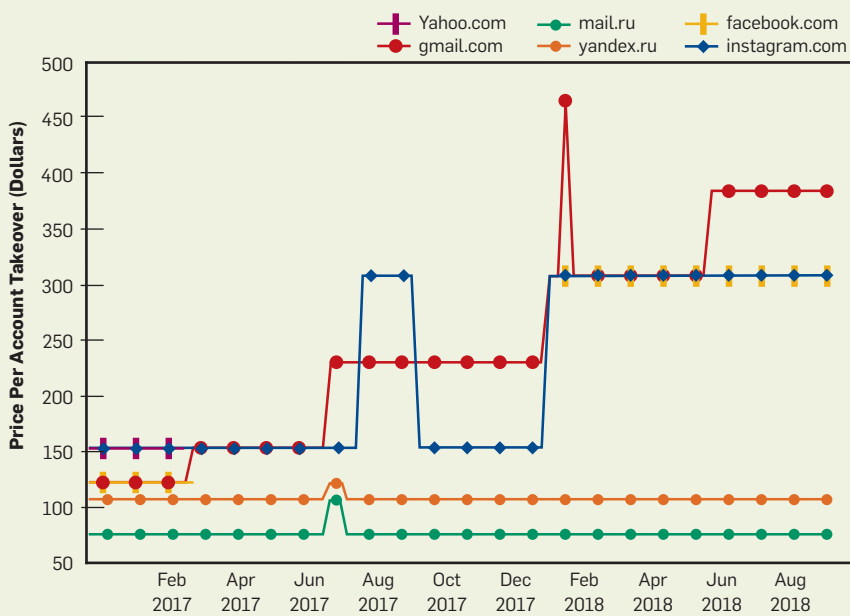
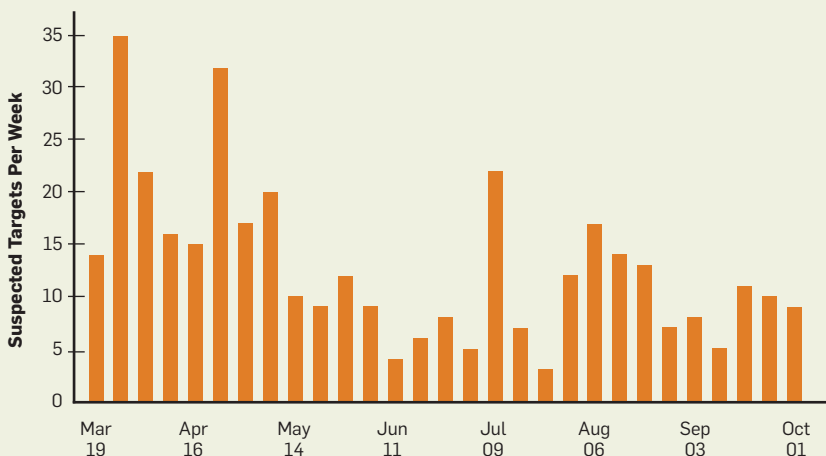


Figure 6. Accounts associated with hack-for-hire services.



count, at-risk populations should use a security key as their 2FA protection, as that code is unphishable.

Post compromise. Upon gaining access to a victim's account, hack-for-hire services start to remove any evidence of compromise and ensure their ability to regain access if needed. Services that gained access to our victim accounts proceeded to sign in to each account and remove all Google email notifications related to a new device sign-in from both the inbox and trash. None of the services changed the password, but we did observe three services remove the 2FA authentication and recovery number from our victim accounts quickly after they gained entry. We presume they took this step to ensure the buyer could gain access to the account and so that the service itself could regain access to the account, but we did not see any service log into the account after the initial login. In essence, the services took precautions to remove their digital footprint from the Google accounts they were breaking into.

Once the email account was accessed, all but one of the services initiated a portability feature in Google called Takeout, allowing them to download the victim account's email content, and provided the parcel of information to the buyer persona. Only one service avoided logging into our victim account and provided the password to the buyer persona without using it first. These findings highlight an emerging risk with data portability and regulations around streamlining access to user data. While intended to improve usability for users, capabilities such as Takeout also increase the ease with which a single hijacking can expose all user data to a service. Since this study, Google has added more step-up verification on sensitive account actions.

Real Victims and Market Activity

Based on our findings from this process, we analyzed the forums of the most successful services to understand their pricing for other services. Moreover, we present an estimate of the number of real victims affected by these services based on login traces from Google. Our findings suggest this market is quite niche.

Alternative services and pricing. While our investigation focused main-

ly on Google—because of legal constraints—many of the hack-for-hire services we interacted with also purported to be capable of breaking into other types of accounts. Figure 4 shows the prices of the hack-for-hire services as of Oct. 10, 2018. All prices are in U.S. dollars, converted from rubles. An * indicates that the service's advertised price was lower than the final payout requested.

Across these five services, hijacking Russian email providers was the least expensive offering, while hijacking a Google or Yahoo account was the most expensive. Breaking into a social media account fell in the middle of these two extremes. The advertisements for one of the services exhibited prices that changed over time, shown in Figure 5. The price of Google account hacking increased the most—from \$123 to \$384 per account over two years—while the cost of Russian email provider hacking has remained largely the same. These differences and price changes are probably the result of a multitude of factors such as demand, changes in security, and competition from other services.

Victims over time. Of the 27 unique services we contracted, only three were able to log into the victim email accounts successfully. Google analyzed associated metadata with the successful login attempts and found that all three services relied on an identical automation process for password validity checks, bypassing any security obstacles such as 2FA, and downloading the victim email archive. While the email sender and delivery addresses differed among the various contracts, the login automation process was the same across the eight months these services were contracted. Google was able to create a signature for this automated login fingerprint and retroactively analyze how many Gmail accounts had a suspicious login attempt.

Google identified 372 accounts targeted by this automated login framework from Mar. 16, 2018, to Oct. 15, 2018, or about one in every one million Google users. Figure 6 shows the weekly breakdown of the number of targeted Google accounts. Be aware that these numbers are lower bounds, since we cannot infer how many users were targeted by these services but did not click on the link (or provide their infor-

mation to grant access), only how many users had an account that was accessed by these services. Despite these limitations, the volume of activity for hack-for-hire services is quite small when compared with other services such as off-the-shelf phishing kits, which impact more than 12 million users a year.¹³ We suspect the hack-for-hire market is small compared with other markets, such as malware distribution.

Discussion

Overall, hack-for-hire services charging \$100–\$400 per contract were found to produce sophisticated, persistent, and personalized attacks able to bypass 2FA via phishing. The demand for these services, however, appears to be limited to a niche market, as evidenced by the small number of discoverable services, an even smaller number of successful services, and the fact that these attackers target only about one in a million Google users. Moreover, this market suffers from poor customer service, as many of the services were slow or inconsistent in their responses to our buyer personas.

Regardless of the behavior of the market, this study sheds light on the importance of security keys for populations who believe they are at risk, as only a security key can protect a user from the attacks viewed in this study. As the market evolves and defenses change, however, attacks might also change and shift from phishing to more persistent threats such as malware.

In conjunction with this study, Google introduced two new defenses to help protect against man-in-the-middle phishing, which in turn would protect against these services. Google now runs additional heuristics when you log in, and also prevents some forms of automated login frameworks. In addition, two of the services have nearly doubled the price of hacking Google accounts since Google rolled out the new protections to users, although it is not known if this price hike is coincidental or was caused by the increased Google protections.

Acknowledgments

Thanks to the co-authors of the original research publication for their feedback in writing this article: Kurt Thomas, Geoffrey M. Voelker, Joe De-

Blasio, and Stefan Savage. Thanks to Mikhail Kolmogorov for his significant assistance, as well as translation help from Kirill Levchenko, Vector Guo Li, and Ivan Mikhailin. And thanks to Shawn Loveland, Elie Bursztein, Angelika Moscicki, Tadek Pietraszek, and Kashyap Puranik. This work was supported in part by NSF grants CNS-1629973 and CNS-1705050, and DHS grant AFRL-FA8750-18-2-0087. 

Related articles on queue.acm.org

Criminal Code

Thomas Wadlow, Vlad Gorelik
<https://queue.acm.org/detail.cfm?id=1180192>

The Seven Deadly Sins of Linux Security

Bob Toxen
<https://queue.acm.org/detail.cfm?id=1255423>

The Web Won't Be Safe or Secure until We Break It

Jeremiah Grossman
<https://queue.acm.org/detail.cfm?id=2390758>

References

1. Anise, O., and Lady, K. State of the auth: Experiences and perceptions of multi-factor authentication. *Duo Security*, 2017; <https://duo.sc/2kmOBid>.
2. Cohen, W.W. Enron email dataset, 2015; <https://www.cs.cmu.edu/~enron/>.
3. Coonce, S. The most expensive lesson of my life: Details of SIM port hack, 2019; <http://bit.ly/2IGSD4Y>.
4. Google. Protect users with the Advanced Protection Program; <https://support.google.com/a/answer/9010419>.
5. Google. Protect your business with 2-Step Verification; <https://support.google.com/a/answer/175197>.
6. Google. Verify a user's identity with extra security; <https://support.google.com/a/answer/6002699>.
7. Honan, M. How Apple and Amazon security flaws led to my epic hacking. *Wired*; <https://www.wired.com/2012/08/apple-amazon-mat-honan-hacking/>.
8. Liu, S., Foster, I., Savage, S., Voelker, G.M., Saul, L.K. Who is .com? Learning to parse WHOIS records. In *Proceedings of the ACM Internet Measurement Conf.*, 2015, 369–380; <https://dl.acm.org/citation.cfm?id=2815675.2815693>.
9. Matishak, M. How Podesta became a cybersecurity poster child. *Politico* 2016; <https://politi.co/2m4fNmd>.
10. Mirian, A., DeBlasio, J., Savage, S., Voelker, G.M., Thomas, K. Hack for hire: Exploring the emerging market for account hijacking. In *Proceedings of the World Wide Web Conf.*, 2019, 1279–1289; <https://dl.acm.org/citation.cfm?id=3313489>.
11. Onaolapo, J., Mariconti, E., Stringhini, G. What happens after you are pwnd: Understanding the use of leaked webmail credentials in the wild. In *Proceedings of the ACM Internet Measurement Conf.*, 2016, 65–79; <https://dl.acm.org/citation.cfm?id=2987475>.
12. Thomas, K. et al. Framing dependencies introduced by underground commoditization. In *Proceedings of the Workshop on the Economics of Information Security*, 2015.
13. Thomas, K. et al. Data breaches, phishing, or malware? Understanding the risks of stolen credentials. In *Proceedings of the ACM Conf. Computer and Communications Security*, 2017, 1421–1434; <https://dl.acm.org/citation.cfm?id=3134067>.

Ariana Mirian is a Ph.D. student in the computer science and engineering department at the University of San Diego, CA, USA, where she focuses on understanding security and privacy via an empirical lens.

Copyright held by author/owner.
Publication rights licensed to ACM.

Application programming interfaces speak louder than words.

BY THOMAS A. LIMONCELLI

API Practices If You Hate Your Customers

DO YOU HAVE disdain for your customers? Do you wish they would go away? When you interact with customers are you silently fantasizing about them switching to your competitor's product? In short, do you hate your customers?

Maybe you should try using your company's external APIs to show your disdain. What? How could you do that?

In this article, I document a number of industry best practices designed to show customers how much you hate them. All of them are easy to implement. Heck, your company may be doing many of them already.

Why would you want to use your company's API to show your hate? I think the answer is quite simple: Customers are jerks.

Darn customers! Always using our services!

Bothering our salespeople for quotes! Creating more work for the accounts receivable department by sending us money. Needing customer support for stupid reasons such as: "The documentation is wrong," or "This feature is broken," or "Your product killed my cat."

See? Jerks.

Older readers may long for the good old days when companies that were actual monopolies would pretend to love their customers. Now we all work for companies that don't admit to being monopolies and actually hate their customers. Boy, how times have changed.

Technique #1: Don't Have an API

Not having an API is a good start. It also requires the least effort of all the techniques. All you have to do is think about adding an API, then *not* do it.

What good is an API, anyway? Primarily it allows customers to implement features that you didn't think of. "Look, buddy, if we didn't think of the feature, it couldn't possibly be very good. We hire the best and brightest to think of new features all day long and not implement them. Don't horn in on their turf."

APIs also permit customers to use a lot more of your product. If they have to click, click, click to use your product, they are going to use it only a little. If an API exists, they can automate their use of your product, which would let them use it a lot more. They could automate provisioning for their entire company. They could build entire new applications based on your API. Just think how much more of your product they would be able to consume with an API.

How totally rude! If they use your product more, you will have to buy more servers, spend more time cashing their checks, and, heaven forbid, maybe start hosting conferences where people use terms such as *leverage*, *hackathons*, and *chalk talks*. Conferences? Ick.

Technique #2: Make Signups Difficult

OK, you have lost the battle and your company wants to build an API anyway. At least you can press the brakes a bit



SO I'M LIKE

“YOUR CALL IS VERY IMPORTANT TO US.”

by requiring a complicated signup process. Self-service onboarding is hedonistic and could lead to dancing.

There are a variety of ways to make onboarding arduous for customers. Some companies require that you open a ticket or speak to an actual human being. That will make any introverted developer think twice before using your API.

Some companies want you to fill out an application form to be able to write an application. Making people beg to use your product is a good way to discourage new users.

For best results, the questions on such forms should be written by someone who previously worked as a CIA interrogator: Why do you want to use this API? What will your application do? Where were you on the night

of the 12th? What's your mother's maiden name? Can you prove she's really your mother?

One such form I filled out required me to describe the application I planned to write. Six months later a SWAT team of auditors appeared at my house, weapons blazing, demanding I show them my code. They wanted to verify I had not lied. If my application didn't match my application, then I could be sent to application jail.

OK, that's not a true story. I did, however, once see that question on a form. Sadly, I didn't have a particular application in mind. I was going to explore the API and write a few simple Python-based utilities to automate some daily tasks. I didn't want to explain all that, however, for fear my

answer would not be good enough for whoever was judging my application. In a panic, I simply described my application as “dark purple with white highlights.” A few weeks later my application was approved. So far, I haven't been visited by any auditor SWAT teams, but as a precaution my code editor has been themed in dark purple with white highlights ever since.

Sadly, some companies do not understand how to make signups difficult. They either make the process entirely self-service, or don't require any kind of signup process at all. When will they ever learn?

Technique #3: Charge Extra. A Lot.

Another way to send customers packing is to charge a lot for your API.

Remember when mobile phone companies charged \$100 for a \$2 data cable? Why can't APIs be like that? You have the potential to inflict a surcharge on anyone who dares to want to make efficient use of your product. Don't squander this opportunity!

It is normal to charge for your service, or include API access only with the enterprise edition. In fact, that's a good way to keep out spammers or others who would abuse the service.

But that's not what I'm talking about here. I mean you should charge a lot extra for the API. A. Lot. Of. Money.

Make it a revenue stream instead of a way to encourage people to use your product. Make API access so expensive that the sales department thinks API stands for additional profit incentive.

This is a lot easier for on-premises software. There the SDK can be sold separately, perhaps using an otherwise unadvertised SKU. Require management approval, a blessing from the pope, and a note from your mother.

Technique #4: Hide the API Docs from Search Engines

Nothing says "We don't actually want you to use our API" like making your API documentation invisible to search engines. The "build, run, debug" cycle of decades ago has been replaced by "run, crash, Google, fix." If your API documentation doesn't appear in search engines, you've sent your customers back to the old days.

Luckily, this can be easily done by putting the documentation behind a login screen. If Google can't crawl it, it can't index it. Googling for answers about your API will be impossible.

Requiring some kind of registration or login to access your API documentation also prevents your competition from examining your API and learning from it. No competitors have ever thought to register using their home address, or to share a password from a friend, right? Never. They are not that smart. It could never possibly happen. You would certainly never do that, so why would they?

If your management refuses to hide documentation behind a login, consider making your documentation a PDF file. This is nearly as frustrating. Most search engines can peer into PDFs, but not if you print the documentation and scan in each page as a bitmap. If search engines OCR such documents, just reformat your text in columns, or tilt the document when you scan it. Be strong, young soldier! With a little elbow grease and a lot of moxie, you can stay one step ahead of anyone who wants to make it easy to access your documentation.

Technique #5: Use a Terrible Protocol Many APIs use JSON:API (<https://jsonapi.org>) or JSON-RPC (www.jsonrpc.org). They are lightweight, easy to use, and easy to debug.

Why would anyone do that? Debugging is boring. Wouldn't you rather appeal to customers who write bug-free code on the first try?

To really show disdain for your customers, use a proprietary protocol so that language support is limited to the client libraries you provide, preferably as binary blobs that are never updated. If you design it carefully, a proprietary protocol can be difficult to understand and impossible to debug, too.

Alternatively, you can use SOAP (Simple Object Access Protocol). According to Wikipedia, SOAP "can be bloated and overly verbose, making it bandwidth-hungry and slow. It is also based on XML, making it expensive to parse and manipulate—especially on mobile or embedded clients" (<https://en.wikipedia.org/wiki/SOAPjr>). Sounds like a win-win!

Technique #6: Permit Only One API Key

One of my favorite ways to show disdain for a customer is to permit only one API key at a time. Anything you can do to make the customer's operations full of toil and cognitive load says, "I don't care about you."

An API key is basically a password that identifies and authenticates a customer. You have only one password for your email account; why would you need more than one? That would be weird, right?

Eventually, customers will need to change, or "rotate," their API key. Maybe it was leaked. Maybe an employee left the company and has a copy of the key. Maybe they just need to rotate the key yearly as part of their security policy.

Here's the hilarious part. If you permit only one API key at a time, you've created a catch-22 situation. Customers can't change the API key on the server, because the clients will lose access until they've been updated, too. They can't change the clients first because the server won't yet know about the new API key. If there are multiple clients, then you're basically expecting your customers to flash-cut all clients at the exact same time. Basic physics says that can't happen. Even if it could happen, there's no way to canary the new key; you've added deployment risk and complexity where nobody would have expected.

What does your API reveal about your feelings toward your customers?		
Technique	Treat customers with disdain	Show customers love
1	Don't have an API	Have an API
2	Make signups difficult, users must justify their request	Self-service onboarding
3	Exorbitant fees for the privilege of API access	Enable API access for free or as part of an "enterprise-level" package
4	API documentation behind login page or otherwise hidden from search engines	API documentation freely accessible and referenced by public search engines
5	Use a proprietary or terrible protocol	Use an industry-standard protocol such as JSON:API or gRPC (https://grpc.io)
6	Permit only one API key	Permit multiple API keys for easy rotation
7	Tempt fate by maintaining documentation manually	Keep documentation in sync with code using automated systems such as Swagger or gRPC
8	Ignore the infrastructure as code (IaC) revolution	Make IaC a top priority: Provide officially supported modules for Terraform, Chef, Puppet, Chocolatey, and similar systems
9	Design APIs to be non-idempotent whenever possible	Design APIs to be idempotent whenever possible

They say the secret to great comedy is timing. Imagine the hilarity of a customer using your product for a year before realizing key rotation is logistically impossible. Surprise!

Hilarity? Or disdain. Whatever.

Some companies do not understand comedy, or how to show disdain for their customers. They permit customers to add a new key on the server, slowly roll out the new key to all clients, testing along the way, then deactivate the old key. Gross!

Technique #7: Maintain Documentation Manually

As your API evolves it is possible for the API and the documentation to get out of sync. Nothing says “I don’t care about my users” like building a system that encourages this kind of error.

Or you can go for the trifecta: an API that is out of sync with the documentation, which is out of sync with the client libraries you provide.

Sure, there are systems such as Swagger (<https://swagger.io/tools/open-source/>) and gRPC (www.grpc.io) that let you define APIs and their documentation in one place, then automatically generate the documentation, server stubs, client SDK bindings in multiple languages, and so on. But what’s the fun in doing work once and letting computers generate all the downstream artifacts you need for free? Consistency is for simpletons.

Technique #8: Ignore The IaC Revolution

The ability to treat infrastructure as code (IaC) is becoming a top priority for operational teams. It not only makes operations easier, more testable, and more reliable, but also paves the path to security compliance best practices required by the likes of SOC2 (Service Organization Controls) and PCI (payment card industry).

Some companies waste their time making it easy for customers to do this. They provide officially supported modules for accessing their services from Terraform, Ansible, Chef, Puppet, and similar systems. They make their client-side software easy to consume by hosting repositories for multiple Linux distributions, and they provide a Chocolatey feed for easy installation on Windows.



**Nothing says
“We don’t
actually want you
to use our API”
like making
your API
documentation
invisible to
search engines.**



It’s much simpler to ignore all of these technologies and hope that the open-source community will provide. Yes, this may result in a confusing array of incompatible options, but you can trumpet the benefits of “user choice.”

Technique #9: Don’t Be Idempotent

I’ve saved the nerdiest technique for last.

An operation is idempotent if performing it multiple times yields the same result as performing it exactly once.

Suppose there’s an API call that creates a virtual machine (VM). If this API call is idempotent, the first time we call it the VM is created. The second time it is called the system detects that the VM already exists and simply returns without error. If this API call is non-idempotent, calling it 10 times will result in 10 VMs being created. (Note: the opposite of *idempotent* isn’t *potent*.)

Similarly, an idempotent delete call will remove the object; subsequent calls will quietly do nothing and return a success status code. If the call were non-idempotent, the second call would return a “not found” error, which would confuse the developers and potentially make them question the meaning of existence.

Why would anyone issue the same API call more than once? When dealing with RPCs (remote procedure calls), the response may be success, failure, or no reply at all. If you don’t hear back from the server, you have to retry the request.

With an idempotent protocol you can simply resend the request. With a non-idempotent protocol, every action must be followed by code that discovers the current state and does the right thing to recover. Putting all that recovery logic in the client is a layering violation.

In the VM example, you would have to query the inventory and see if the VM you asked to create exists. If it does exist, you must make sure it was created properly or is in a good state. If it is in a bad state, you repair it or delete it and start over. The list of potential conditions and edge cases goes on and on.

That was a simple example. Recovery from other API calls can be even more complex. The attempts to recover from failures may also fail. Now you are faced with an infinitely recursive world of failures, failed re-

covery attempts, and on and on. Code that looks sane on first glance ends up creating zero VMs, or three VMs, or more. With multiple simultaneous clients, you must deal with timing, locking problems, crossed messages, and a nest of heisenbugs.

Putting this logic in the client library ensures the client will need more frequent updating. Requiring the user to implement the recovery logic is delightfully evil: how would they even know what they should implement?

These problems are reduced or eliminated when the API is idempotent.

Why not simply use a more reliable network? Oh, that's just adorable. Networks are never reliable. They can't be. Thinking that networks are reliable is the first fallacy of distributed computing (https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing).

If networks are unreliable, then a network API is inherently unreliable, too. A request can be lost on its way to the server and never executed. Execution may be complete, but the reply back to us gets lost. The server may reboot during the operation. The client might reboot while sending the request, while waiting for the request, or after receiving the request but before local state is stored on stable storage. So many edge cases!

In distributed computing everything can fail. If you hate your customers, you can make sure that dealing with failure is burdensome, error prone, and just plain impossible to get 100% right. Customers will always be fixing edge cases instead of doing productive work.

Don't spoil the fun. Show your disdain for customers with non-idempotent APIs.

Summary. The accompanying table includes a summary of these techniques along with ways that companies may accidentally provide good service to their API customers.

Getting buy-in. Your coworkers may resist some of these techniques. How do you get them on board?

You could have them read this article, although that could backfire. If the wrong person reads it, he or she might push back and do the opposite.

If that happens, you might end up with a great API that is easy to get started with, easy to use, has great docu-



If networks are unreliable, then a network API is inherently unreliable, too. A request can be lost on its way to the server and never executed.



mentation that is easy to access, and helps people write code that works the first time and every time.

Shirley, You Can't Be Serious!

This article is written in jest to make a point. Although some companies do the bad things set forth here, they don't do them to hurt customers. In my experience, engineers take pride in doing good work and impressing customers with well-made systems. I trust that when companies do the naughty things in this article, it is out of ignorance, lack of resources, or an impossible deadline.

Luckily, in some cases the good practice is easier to implement than the bad practice. Creating an authentication system to restrict access to documentation is more difficult than making the documentation freely available. Putting all documentation on one long page so that it can be searched using Ctrl-F is easier than putting each API call on a separate page.

Sadly, some of these good practices do require a lot of work. Creating a self-service onboarding system is not easy. It requires usability testing and revisions. Ease of use is never achieved on the first guess.

Justifying the resources required for all these good practices may be difficult, especially when an API isn't used by many of your customers. "What's the ROI when hardly anyone uses our API?" your management may ask. I look at it differently: Maybe usage is low because you haven't done these things. 

Related articles on queue.acm.org

Programmers are People, Too

<https://queue.acm.org/detail.cfm?id=1071731>

Forked Over

Kode Vicious

<https://queue.acm.org/detail.cfm?id=2611431>

Managing Technical Debt

Eric Allman

<https://queue.acm.org/detail.cfm?id=2168798>

Thomas A. Limoncelli is the SRE manager at Stack Overflow Inc. in New York City. His books include *The Practice of System and Network Administration*, *The Practice of Cloud System Administration*, and *Time Management for System Administrators*. He blogs at EverythingSysadmin.com and tweets at [@YesThatTom](https://twitter.com/YesThatTom).

Copyright held by author/owner.
Publication rights licensed to ACM.

Concurrency

The Works of Leslie Lamport

This book is a celebration of Leslie Lamport's work on concurrency, interwoven in four-and-a-half decades of an evolving industry: from the introduction of the first personal computer to an era when parallel and distributed multiprocessors are abundant. His works lay formal foundations for concurrent computations executed by interconnected computers. Some of the algorithms have become standard engineering practice for fault tolerant distributed computing - distributed systems that continue to function correctly despite failures of individual components. He also developed a substantial body of work on the formal specification and verification of concurrent systems, and has contributed to the development of automated tools applying these methods.

Part I consists of technical chapters of the book and a biography. The technical chapters of this book present a retrospective on Lamport's original ideas from experts in the field. Through this lens, it portrays their long-lasting impact. The chapters cover timeless notions Lamport introduced: the Bakery algorithm, atomic shared registers and sequential consistency; causality and logical time; Byzantine Agreement; state machine replication and Paxos; temporal logic of actions (TLA). The professional biography tells of Lamport's career, providing the context in which his work arose and broke new grounds, and discusses LaTeX - perhaps Lamport's most influential contribution outside the field of concurrency. This chapter gives a voice to the people behind the achievements, notably Lamport himself, and additionally the colleagues around him, who inspired, collaborated, and helped him drive worldwide impact. Part II consists of a selection of Leslie Lamport's most influential papers.

This book touches on a lifetime of contributions by Leslie Lamport to the field of concurrency and on the extensive influence he had on people working in the field. It will be of value to historians of science, and to researchers and students who work in the area of concurrency and who are interested to read about the work of one of the most influential researchers in this field.

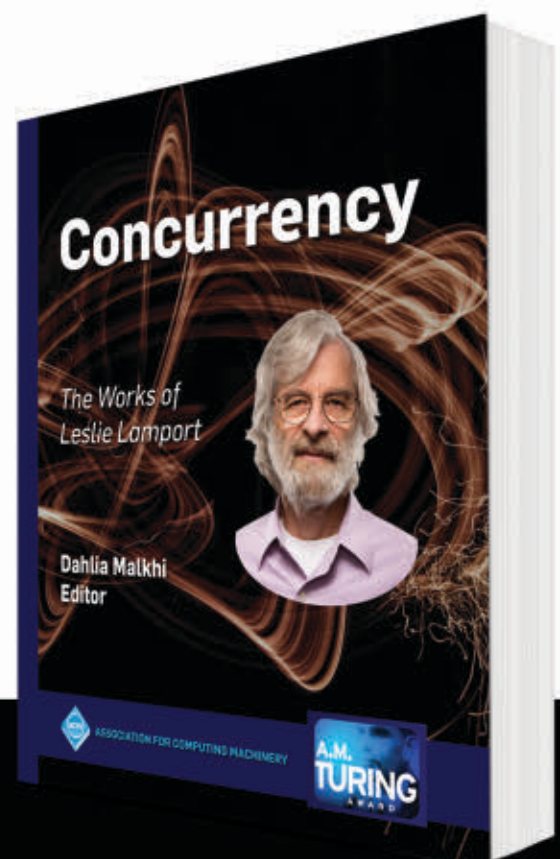
Dahlia Malkhi, Editor

ISBN: 978-1-4503-7271-8

DOI: 10.1145/3335772

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



ACM BOOKS
Collection II

DOI:10.1145/3368454

The server is dead, long live the server.

BY PAUL CASTRO, VATCHE ISHAKIAN,
VINOD MUTHUSAMY, AND ALEKSANDER SLOMINSKI

The Rise of Serverless Computing

CLOUD COMPUTING in general, and Infrastructure-as-a-Service (IaaS) in particular, have become widely accepted and adopted paradigms for computing with the offerings of virtual machines (VM) on demand. By 2020, 67% of enterprise IT infrastructure and software spending will be for cloud-based offerings.¹⁶

A major factor in the increased adoption of the cloud by enterprise IT was its pay-as-you-go model where a customer pays only for resources leased from the cloud provider and have the ability to get as many resources as needed with no up-front cost (elasticity).² Unfortunately, the burden of scaling was left for developers and system designers that typically used overprovisioning techniques to handle sudden surges in service requests. Studies of reported usage of cloud resources in datacenters¹⁹ show a substantial gap between the resources that cloud customers allocate and pay for (leasing VMs), and actual resource utilization (CPU, memory, and so on).

Serverless computing is emerging as a new and compelling paradigm for the deployment of cloud

applications, largely due to the recent shift of enterprise application architectures to containers and microservices.²³ Using serverless gives pay-as-you-go without additional work to start and stop server and is closer to original expectations for cloud computing to be treated like as a utility.² Developers using serverless computing can get cost savings and scalability without needing to have a high level of cloud computing expertise that is time-consuming to acquire.

Due to its simplicity and economical advantages, serverless computing is gaining popularity as reported by the increasing rate of the “serverless” search term by Google Trends. Its market size is estimated to grow to 7.72 billion by 2021.¹⁰ Most prominent cloud providers including Amazon, IBM, Microsoft, Google, and others have already released serverless computing capabilities with several additional open source efforts driven by both industry and academic institutions (for example, see CNCF Serverless Cloud Native Landscape^a).

From the perspective of an IaaS customer, the serverless paradigm shift presents both an opportunity and a risk. On the one hand, it provides developers with a simplified programming model for creating cloud applications that abstracts away most, if

a <https://s.cncf.io/>

>> key insights

- **Serverless computing takes the original promises of cloud computing and delivers true pay only for resources used with almost infinite scalability while hiding the details of how servers are used and maintained.**
- **Serverless computing is a new cloud computing paradigm with enormous economic growth potential.**
- **Serverless computing allows the developer to focus on developing business logic and gives the cloud provider additional control over optimizing resources.**
- **There are many technical challenges and opportunities for research.**

ILLUSTRATION BY PETER BOLLINGER

The server
is dead,

long live

the server

not all, operational concerns. They no longer have to worry about availability, scalability, fault tolerance, over/underprovisioning of VM resources, managing servers, and other infrastructure issues. Instead, they can focus on the business aspects of their applications. The paradigm also lowers the cost of deploying cloud code by charging for execution time rather than resource allocation. On the other hand, deploying such applications in a serverless platform is challenging and requires relinquishing design decisions to the platform provider that concern, among other things, quality-of-service (QoS) monitoring, scaling, and fault-tolerance properties. There is a risk an application's requirements may evolve to conflict with the capabilities of the platform.

From the perspective of a cloud provider, serverless computing is an additional opportunity to control the entire development stack, reduce operational costs by efficient optimization and management of cloud resources, offer a platform that encourages the use of additional services in their ecosystem, and lower the effort required to author and manage cloud-scale applications.

Defining Serverless Computing

Serverless computing can be defined by its name—less thinking (or caring) about servers. Developers do not need to worry about low-level details of servers management and scaling, and only pay for when processing requests or events. We define serverless as follows:

Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running.

This definition captures the two key features of serverless computing:

- **Cost**—*billed only for what is running (pay-as-you-go)*. As servers and their usage is not part of serverless computing model, then it is natural to pay only when code is running and not for idle servers. As execution time may be short, then it should be charged in fine-grained time units (like hundreds of milliseconds) and developers do not need to pay for overhead of servers creation or destructions (such as VM booting time).



One of the major challenges slowing the adoption of serverless is the lack of tools and frameworks.



This cost model is very attractive to workloads that must run occasionally; serverless essentially supports “scaling to zero” and avoid need to pay for idle servers. The big challenge for cloud providers is the need to schedule and optimize cloud resources.

- **Elasticity**—*scaling from zero to “infinity.”* Since developers do not have control over servers that run their code, nor do they know the number of servers their code runs on, decisions about scaling are left to cloud providers. Developers do not need to write auto-scaling policies or define how machine-level usage (CPU, memory, and so on) translates to application usage. Instead they depend on the cloud provider to automatically start more parallel executions when there is more demand for it. Developers also can assume the cloud provider will take care of maintenance, security updates, availability and reliability monitoring of servers.

Serverless computing today typically favors small, self-contained units of computation to make it easier to manage and scale in the cloud. A computation, which can be interrupted or restarted, cannot depend on the cloud platform to maintain its state. This inherently influences the serverless computing programming models. There is, however, no equivalent notion of scaling to zero when it comes to state, since a persistent storage layer is needed. However, even if the implementation of a stateful service requires persistent storage, a provider can offer a pay-as-you-go pricing model that would make state management serverless. We are seeing providers releasing services that stretch the definition of serverless, and the definition may evolve over time. For example, Amazon Aurora is a “serverless” database service, which supports powerful auto-scaling capabilities but requires minimum memory and CPU allocations and hence does not scale to zero and has ongoing costs.^b

The most natural way to use serverless computing is to provide a piece of code (function) to be executed by the serverless computing platform. It leads to the rise of Function-as-a-service (FaaS) platforms focused on

^b <https://aws.amazon.com/rds/aurora/serverless/>

allowing small pieces of code represented as functions to run for limited amount of time (at most minutes), with executions triggered by events or HTTP requests (or other triggers), and not allowed to keep persistent state (as function may be restarted at any time). By limiting time of execution and not allowing functions to keep persistent state FaaS platforms can be easily maintained and scaled by service providers. Cloud providers can allocate servers to run code as needed and can stop servers after functions finish as they run for limited amount of time. If functions must maintain state, then they can use external services to persist their state.

FaaS is an embodiment of serverless computing principles, which we define as follows:

Function-as-a-Service is a serverless computing platform where the unit of computation is a function that is executed in response to triggers such as events or HTTP requests.

Our approach to defining serverless is consistent with emerging definitions of serverless from industry. For example, Cloud Native Computing Foundation (CNCF) defines serverless computing¹¹ as “the concept of building and running applications that do not require server management. It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.” While our definition is close to the CNCF definition, we make a distinction between serverless computing and providing functions as a unit of computation. As we discuss in the research challenges section, it is possible that serverless computing will expand to include additional aspects that go beyond today’s relatively restrictive stateless functions into possibly long-running and stateful execution of larger compute units. However, today serverless and FaaS are often used interchangeably as they are close in meaning and FaaS is the most popular type of serverless computing.

Paul Johnston (co-founder of ServerlessDays) defined serverless as

follows:^c “A serverless solution is one that costs you nothing to run if nobody is using it (excluding data storage).” This definition highlights the most important characteristic of serverless computing—pays-as-you-go. It assumes serverless computing is a subset of cloud computing so auto-scaling is included and developers have no access to servers. CNCF and our definitions emphasize not only pay-as-you-go or “scale to zero” aspects, but also the lack of need to manage servers.

Another way to define serverless computing is by what functionality it enables. Such an approach emphasizes “serverless is really about the managed services” and FaaS can be treated as cloud “glue,” as described by Steven Faulkner (a senior software engineer at LinkedIn^d). It is “glue” that joins applications composed of cloud services. Such a definition addresses only a narrow set of use cases where serverless computing is used, while our definition captures the important use cases, which we will highlight in the accompanying sidebars.

All definitions share the observation that the name ‘serverless computing’ does not mean servers are not used, but merely that developers can leave most operational concerns of managing servers and other resources, including provisioning, monitoring, maintenance, scalability, and fault-tolerance to the cloud provider.

History and Related Work

The term ‘serverless’ can be traced to its original meaning of not using servers and typically referred to peer-to-peer (P2P) software or client-side only solutions.²⁸ In the cloud context, the current serverless landscape was introduced during an AWS re:Invent event in 2014.³ Since then, multiple cloud providers, industrial, and academic institutions have introduced their own serverless platforms. Serverless seems to be the natural progression following recent advancements and adoption of VM and container technologies, where each step up the abstraction layers led to more lightweight units of computation in terms of resource consumption, cost, and speed of development and

deployment. Furthermore, serverless builds upon long-running trends and advances in both distributed systems, publish-subscribe systems, and event-driven programming models,¹² including actor models,¹ reactive programming,⁴ and active database systems.²⁵

Serverless platforms can be considered an evolution of Platform-as-a-Service (PaaS) as provided by platforms such as Cloud Foundry, Heroku, and Google App Engine (GAE). PaaS was defined by NIST^e as “the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.” In this definition, users are expected to manage deployments of applications and have control over hosting environment configurations.

Serverless FaaS, when compared to this definition of PaaS, is removing user control over hosting to provide simpler scaling and more attractive billing model: the cloud provider controls the hosting environment’s configuration, runs user-provided code only when it is invoked, and only bills for actual usage while hiding the complexity of scaling (in practice implementing auto-scaling in PaaS is not easy and it is very difficult to scale to zero). That is a significant change when compared to the previous generation of PaaS (which could be considered first generation of PaaS) and it is very attractive for PaaS users that do not need to pay for idle resources and avoid managing auto-scaling rules.

The main differentiators of serverless platforms is transparent autoscaling and fine-grained resource charging only when code is running. That should not to be confused with free-usage quota, where limited monthly resource quota is available, but counted even if the application is not used. For example, GAE Standard is priced in “instance hours”^f and even if the app is

c <http://bit.ly/2G3Hp1R>

d <http://bit.ly/2xzNEWB>

e <http://bit.ly/2lXCgkI>

f <http://bit.ly/2kuqZbh>

Table 1. Comparison of different choices for cloud as a service.

	IaaS	1st Gen PaaS	FaaS	BaaS/SaaS
Expertise required	High	Medium	Low	Low
Developer Control/Customization allowed	High	Medium	Low	Very low
Scaling/Cost	Requires high-level of expertise to build auto-scaling rules and tune them	Requires high-level of expertise to build auto-scaling rules and tune them	Auto-scaling to work load requested (function calls), and only paying for when running (scale to zero)	Hidden from users, limits set based on pricing and QoS
Unit of work deployed	Low-level infrastructure building blocks (VMs, network, storage)	Packaged code that is deployed and running as a service	One function execution	App-specific extensions
Granularity of billing	Medium to large granularity: minutes to hours per resource to years for discount pricing	Medium to large granularity: minutes to hours per resource to years for discount pricing	Very low granularity: hundreds of milliseconds of function execution time	Large: typically, subscription available based on maximum number of users and billed in months

not used the instance is kept running. Later, GAE added Flexible version with a more fine-grained billing unit, but still developers will be billed even if server is not used. That can lead to unexpected outcomes when the bill arrives at the end of month for forgotten test services.^g

Mobile Backend as-a-Service (MBaaS) or more generalized Backend as-a-Service (BaaS) bears a close resemblance to serverless computing. Some of those services even provided “cloud functions” (for example, Facebook’s now-defunct Parse Cloud Code). Such

g <https://stackoverflow.com/questions/47125661/>

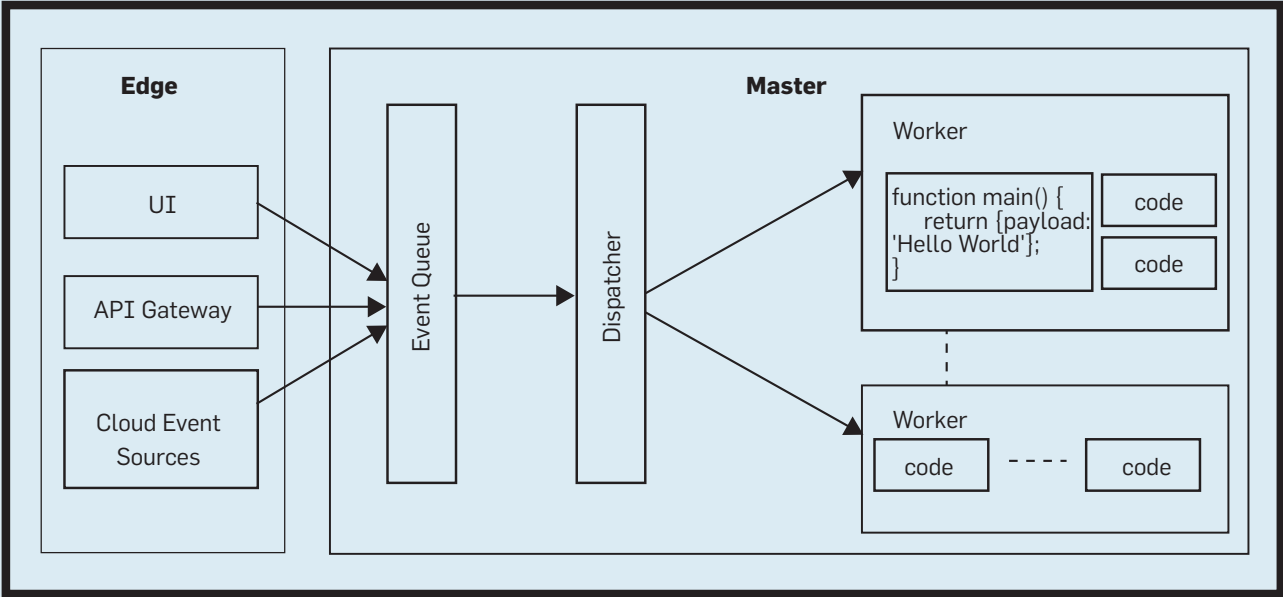
code, however, was typically limited to mobile use cases.

Software-as-a-Service (SaaS) may support the server-side execution of user-provided functions, but they are executing in the context of an application and hence limited to the application domain. Some SaaS vendors allow the integration of arbitrary code hosted somewhere else and invoked via an API call. For example, this is approach is used by the Google Apps Marketplace in Google Apps for Work.

The boundaries defining serverless computing functionality overlaps with PaaS and SaaS. One way to categorize serverless is to consider the varying lev-

els of developer control over the infrastructure. In an IaaS model, the developer has much more control over the resources, but is responsible for managing both the application code and operating the infrastructure. This gives the developer great flexibility and the ability to customize every aspect of the application and infrastructure, such as administering VMs, managing capacity and utilization, sizing the workloads, achieving fault tolerance and high availability. PaaS abstracts away VMs and takes care of managing underlying operating systems and capacity, but the developer is responsible for the full life cycle of the code that is de-

Figure 1. High-level serverless FaaS platform architecture.



ployed and run by the platform, which does not scale down to zero. SaaS represent the other end of the spectrum where the developer has no control over the infrastructure, and instead get access to prepackaged components. The developer is allowed to host code there, though that code may be tightly coupled to the platform. BaaS is similar to SaaS in that the functionality is targeting specific use cases and components, for example, MBaaS provide backend functionality needed for mobile development such as managing push notifications, and when it allows developer to run code it is within that backend functionality (see Table 1).

Architecture

The core functionality of a FaaS framework is simply that of an event processing system, as shown in Figure 1. The service manages a set of user defined functions (a.k.a actions). Once a request is received over HTTP from an event data source (a.k.a. triggers), the system determines which action(s) should handle the event, create a new container instance, send the event to the function instance, wait for a response, gather execution logs, make the response available to the user, and stop the function when it is no longer needed.

The abstraction level provided by FaaS is unique: a short-running stateless function. This has proven to be both expressive enough to build useful applications, but simple enough to allow the platform to autoscale in an application agnostic manner.

While the architecture is relatively simple, the challenge is to implement such functionality while considering metrics such as cost, scalability, latency, and fault tolerance. To isolate the execution of functions from different users in a multitenant environment, container technologies,⁹ such as Docker, are often used.

Upon the arrival of an event, the platform proceeds to validate the event ensuring it has the appropriate authentication and authorization to execute. It also checks the resource limits for that particular event. Once the event passes validation, the platform the event is queued to be processed. A worker fetches the request, allocates the appropriate container, copies over

Figure 2. A function written in JavaScript.

```
function main(params, context) {
  return {payload: 'Hello, ' + params.name
    + ' from ' + params.place};
}
```

Table 2. Real-world applications that use serverless computing.

Where is serverless used?	What do they use serverless computing for?
Aegex	Xamarin application that customers can use to monitor real-time sensor data from IoT devices. ^a
Abilisense	Manages an IoT messaging platform for people with hearing difficulties. They estimated they could handle all the monthly load for less than \$15 a month. ^b
A Cloud Guru	Uses functions to perform protected actions such as payment processing and triggering group emails. In 2017 they had around 200K users and estimated \$0.14 to deliver video course to a user. ^c
Coca-Cola	Serverless Framework is a core component of The Coca-Cola Company's initiative to reduce IT operational costs and deploy services faster. ^d One particular use case is the use of serverless in their vending machine and loyalty program, which managed to have 65% cost savings at 30 million hits per month. ^e
Expedia	Expedia did "over 2.3 billion Lambda calls per month" back in December 2016. That number jumped 4.5 times year-over-year in 2017 (to 6.2 billion requests) and continues to rise in 2018. ^f Example applications include integration of events for their CI/CD platforms, infrastructure governance and autoscaling. ^g
Glucon	Serverless mobile backend to reduce client app code size and avoid disruptions. ^h
Heavywater Inc	Runs Website and training courses using serverless (majority of cost per user is not serverless but storage of video). Serverless reduced their costs by 70%. ⁱ
iRobot	Backend for iRobot products. ^j
Postlight	Mercury Web Parser is a new API from Postlight Labs that extracts meaningful content from Web pages. Serving 39 million requests for \$370/month, or: How We Reduced Our Hosting Costs by Two Orders of Magnitude. ^k
PyWren	Map-reduce style framework for highly parallel analytics workloads. ^l
WeatherGods	A mobile weather app that uses serverless as backend. ^m
Santander Bank	Electronic check processing. Less than \$2 to process all paper checks within a year. ⁿ
Financial Engines	Mathematical calculations for evaluation and optimization of investment portfolios. 94% savings on cost approximately 110K annually. ^o

a <http://bit.ly/2XCas2J>

b <https://thenewstack.io/ibms-openwhisk-serverless/>

c <https://gotochgo.com/2017/sessions/61>

d <http://bit.ly/2xCWYsO>

e <https://aws.amazon.com/serverless/videos/video-lambda-coca-cola/>

f <http://bit.ly/2JktVAR>

g https://www.youtube.com/watch?v=gT9x9LnU_rE

h <http://bit.ly/2lyLaoOs>

i <http://bit.ly/2xzYE6t>

j <https://aws.amazon.com/solutions/case-studies/irobot/>

k <http://bit.ly/2YQcUni>

l <http://pywren.io/>

m <https://thenewstack.io/ibms-openwhisk-serverless/>

n <http://bit.ly/2YHtsxY>

o <https://aws.amazon.com/solutions/case-studies/financial-engines/>

Use Case 1: Event Processing

One class of applications that exemplify the use of serverless is event-based programming. The following use case shows an example of a bursty, compute-intensive workload popularized by AWS Lambda, and has become the “Hello, World” of serverless computing. It is a simple image-processing event handler function.

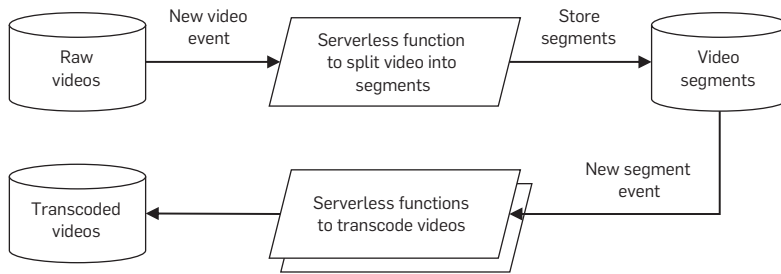
Netflix uses serverless functions to process video files (<https://amzn.to/2xMtwAt>).

The videos are uploaded to Amazon S3,²³ which emits events that trigger Lambda functions that split the video and transcode them in parallel to different formats. The flow is depicted in the figure here.

The function is completely stateless and idempotent, which has the advantage that in the case of failure (such as network problems accessing the S3 folder), the function can be executed again with no side effects.

While the example here is relatively simple, by combining serverless functions with other services from the cloud provider, more complex applications can be developed, for example, stream processing, filtering and transforming data on the fly, chatbots, and Web applications.

Video processing.



the function—use code from storage use—into the container and executes the event. The platform also manages stopping and deallocating resources for idle function instances.

Creating, instantiating, and destroying a new container for each function invocation while can be expensive and introduces an overall latency, which is referred to as the *cold start problem*. In contrast, warm containers are containers that were already instantiated and executed a function. Cold start problems can be mitigated by techniques such as maintaining a pool of uninstantiated stem cell containers, which are containers that have been previously instantiated but not assigned to a particular user, or reuse a warm container that have been previously invoked for the same user.⁷ Another factor that can affect the latency is the reliance of the user function on particular libraries (for example, numpy) that must be downloaded and installed before function invocation. To reduce startup time of

cloud functions, one can appropriately cache the most important packages across the node workers, thus leading to reduced startup times.²⁴

In typical serverless cloud offerings, the only resource configuration customers are allowed to configure is the size of main memory allocated to a function. The system will allocate other computational resources (for example, CPU) in proportion to the main memory size. The larger the size, the higher the CPU allocation. Resource usage is measured and billed in small increments (for example, 100ms) and users pay only for the time and resources used when their functions are running.

Several open source serverless computing frameworks are available from both industry and academia (for example, Kubeless, OpenLambda, OpenWhisk, OpenFaaS). In addition, major cloud vendors such as Amazon, IBM, Google, and Microsoft have publically available commercial server-

less computing frameworks for their consumers. While the general properties (for example, memory, concurrent invocations, maximum execution duration of a request) of these platforms are relatively the same, the limits as set by each cloud provider are different. Note the limits on these properties are a moving target and are constantly changing as new features and optimizations are adopted by cloud providers. Evaluating the performance of different serverless platforms to identify the trade-offs has been a recent topic of investigation,^{17,20,26} and recent benchmarks have been developed to compare the serverless offering by the different cloud providers.^h

Programming Model

A typical FaaS programming model consists of two major primitives: *Action* and *Trigger*. An Action is a stateless function that executes arbitrary code. Actions can be invoked asynchronously in which the invoker—caller request—does not expect a response, or synchronously where the invoker expects a response as a result of the action execution. A Trigger is a class of events from a variety of sources. Actions can be invoked directly via REST API, or executed based on a trigger. An event can also trigger multiple functions (parallel invocations), or the result of an action could also be a trigger of another function (sequential invocations). Some serverless frameworks provide higher-level programming abstractions for developers, such as function packaging, sequencing, and composition, which may make it easier to construct more complex serverless apps.

Currently, serverless frameworks execute a single main function that takes a dictionary (such as a JSON object) as input and produces a dictionary as output. They have limited expressiveness as they are built to scale. To maximize scaling, serverless functions do not maintain state between executions. Instead, the developer can write code in the function to retrieve and update any needed state. The function is also able to access a context object that represents the environment in which the function is running (such as a security context). As shown in Figure 2, a func-

^h <http://faasmark.com/>

tion written in JavaScript could take as input a JSON object as the first parameter, and context as the second.

Current Cloud Provider Serverless offerings support a wide variety programming languages, including Java, Python, Swift, C#, and Node.js. Some of the platforms also support extensibility mechanisms for code written in any language as long as it is packaged in a Docker image that supports a well-defined API.

Due to the limited and stateless nature of serverless functions, and its suitability for composition of APIs, cloud providers are offering an ecosystem of value added services that support the different functionalities a developer may require, and is essential for production ready applications. For example, a function may need to retrieve state from permanent storage, such as a file server or database, another may use a machine learning service to perform some text analysis or image recognition. While the functions themselves may scale due to the serverless guarantees, the underlying storage system itself must provide reliability and QoS guarantees to ensure smooth operation.

Tools and Frameworks

One of the major challenges slowing the adoption of serverless is the lack of tools and frameworks. The tools and frameworks currently available can be categorized as follows: development, testing, debugging, deployment. Several solutions been proposed to deal with these categories.

Almost all cloud providers provide a cloud-based IDE, or extensions/plugins to popular IDEs that allows the developer to code and deploy serverless functions. They also provide a local containerized environment with an SDK that allows the developer to develop and test locally serverless functions before deploying it in a cloud setting. To enable debugging, function execution logs are available to the developer and recent tools such as AWS X-Rayⁱ allow developers to detect potential causes of the problem.²² Finally, there are open source frameworks^j that allow developers to define serverless func-

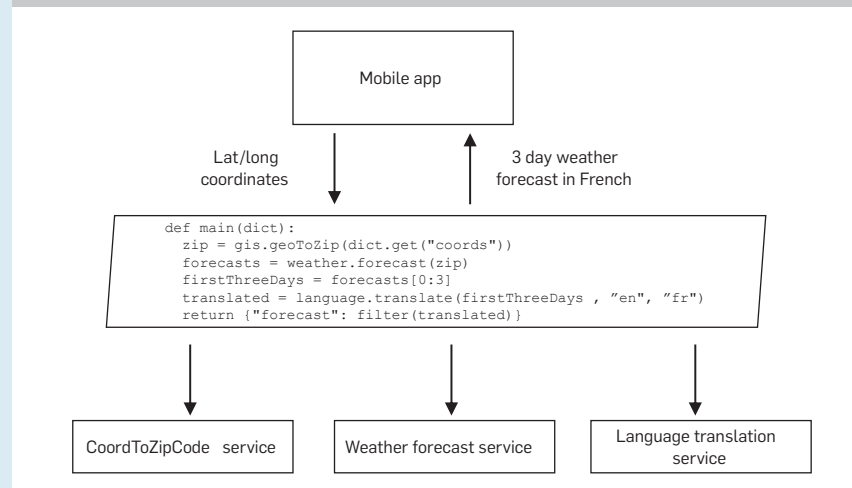
i <https://aws.amazon.com/xray/>

j <https://serverless.com/>

Use Case 2: API Composition

Consider a mobile app that sequentially invokes a geolocation, weather, and language translation APIs to render the weather forecast for a user's current location. A short serverless function can be used to invoke these APIs. Thus the mobile app avoids invoking multiple APIs over a potentially resource constrained mobile network connection, and offloads the filtering and aggregation logic to the backend. Glucon, for example, used serverless in its conference scheduler application to minimize client code, and avoid disruptions.

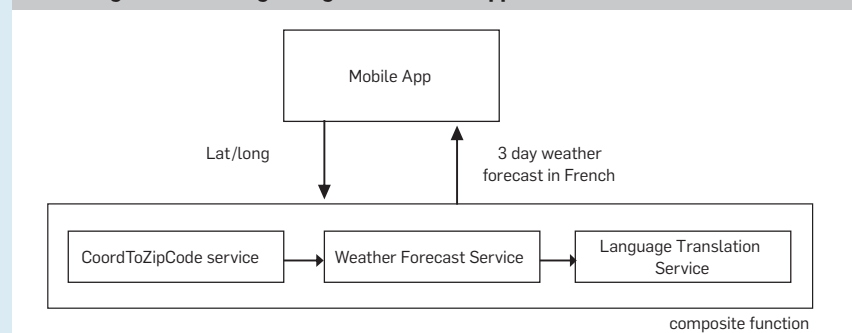
A serverless anti-pattern of offloading API calls from mobile app to backend.



Note the main function is acting as an orchestrator that is waiting for a response from a function before invoking another, thus incurring a cost of execution while the function is basically waiting for I/O. Such a pattern of programming is referred to as a serverless anti-pattern.

The serverless programming approach would be to encapsulate each API call as serverless function, and the chain the invocation of these functions in a sequence. The sequence itself behaves as a composite function.

Offloading API calls and glue logic from mobile app to backend.



More complex orchestrations can use technologies like AWS Step Functions and IBM Composer to prevent serverless anti-patterns but may incur additional costs due to the services.

tions, triggers, and services needed by the functions. These frameworks will handle the deployment of these functions to the cloud provider.

Use Cases

Serverless computing has been utilized to support a wide range of ap-

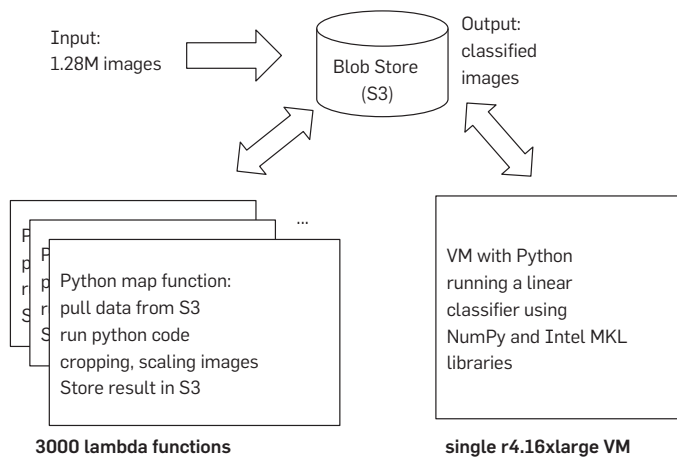
plications. From an infrastructure perspective, serverless and more traditional architectures may be used interchangeably or in combination. The determination of when to use serverless will likely be influenced by other non-functional requirements, such as the amount of control over operations

Use Case 3: Map-Reduce Style Analytics

PyWren¹⁸ (illustrated in the figure) is a Python-based system that utilizes the serverless framework to help users avoid the significant development and management overhead of running MapReduce jobs. It is able to get up to 40TFLOPS peak performance from AWS Lambda, using AWS S3 for storage and caching. A similar reference architecture has been proposed by AWS Labs (<https://github.com/aws-labs/lambda-refarch-mapreduce>).

PyWren exemplifies a class of use cases that uses a serverless platform for highly parallel analytics workloads.

Map + monolithic Reduce PyWren example implementing ImageNet Large Scale Visual Recognition Challenge.



required, cost, as well as application workload characteristics.

From a cost perspective, the benefits of a serverless architecture are most apparent for bursty^{5,27} workloads. Bursty workloads fare well because the developer offloads the elasticity of the function to the platform, and just as important, the function can scale to zero, so there is no cost to the consumer when the system is idle.

There are many areas where serverless computing is used today. Table 2 provides a representative list of different types of applications used in different domains along with a short description. We emphasize this list is not exhaustive; we offer it to identify and discuss emerging patterns. Interested readers can find examples by going through additional use cases that are publically available by cloud providers.

From a programming model perspective, the stateless nature of serverless functions lends themselves to application structure similar to those found in functional reactive program-

ming. This includes applications that exhibit event-driven and flow-like processing patterns (see Use Case 1 sidebar of thumbnail creation).

As a comparison, consider an equivalent solution implemented as an application running on a set of provisioned VMs. The logic in the application to generate the thumbnails is relatively straightforward, but the user must manage the VMs, including monitoring traffic loads, auto-scaling the application, and managing failures. There is also a limit to how quickly VMs can be added in response to bursty workloads, forcing the user to forecast workload patterns and pay for pre-provisioned resources. The consequence is there will always be idle resources, and it is impossible to scale down to zero VMs. In addition, there must be a component that monitors for changes to the S3 folder, and dispatch these change events to one of the application instances. This dispatcher itself must be fault-tolerant and auto-scale.

Another class of applications that exemplify the use of serverless is composition of a number of APIs, controlling the flow of data between two services, or simplify client-side code that interacts by aggregating API calls (see Use Case 2 sidebar).

Serverless computing may also turn out to be useful for scientific computing. Having ability to run functions and not worry about scaling and paying only for what is used can be very good for computational experiments. One class of applications that started gaining momentum are compute intensive applications.¹³ Early results show (see Use Case 3 sidebar) the performance achieved is close to specialized optimized solutions and can be done in an environment that scientists prefer such as Python.

If the workloads cannot be easily divided into smaller units (such as Python functions), then batch-oriented systems such as high-performance computing (HPC) or MapReduce clusters are a better option. If the demand for such clusters can be sustained, for example, by having job queues where jobs are submitted and scheduled based on available resources, then workloads can be executed more cheaply, albeit possibly taking longer to complete. The cost is lower than using FaaS as the service provider can get cheaper VMs either by buying actual servers, using vendor platforms such as Databricks or BigQuery, or getting reserved VMs with longer contracts. If batch workloads can tolerate occasional restarts it may be better to run such workloads with on-demand VMs (such as AWS spot instances).

Many “born in cloud” companies build their services to take full advantage of cloud services. Whenever possible they use existing cloud services and built their functionality using serverless computing. Before serverless computing they would need to use virtual machines and create auto-scaling policies. Serverless computing, with its ability to scale to zero and almost infinite on-demand scalability, allows them to focus on putting business functionality in serverless functions instead of becoming experts in low-level cloud infrastructure and server management (see Use Case 4 sidebar for more details).

Challenges and Limitations

Serverless computing is a large step forward, and is receiving a lot of attention from industry and is starting to gain traction among academics. Changes are happening rapidly and we expect to see different evolutions of what is serverless and FaaS. While there are many immediate innovation needs for serverless,^{6,14,15} there are significant challenges that need to be addressed to realize full potential to serverless computing. Based on discussions during a series serverless workshops organized by the authors (<https://www.serverlesscomputing.org/workshops/>), and several academic²¹ and industrial surveys (<https://www.digitalocean.com/currents/june-2018/>), we outline the following challenges:

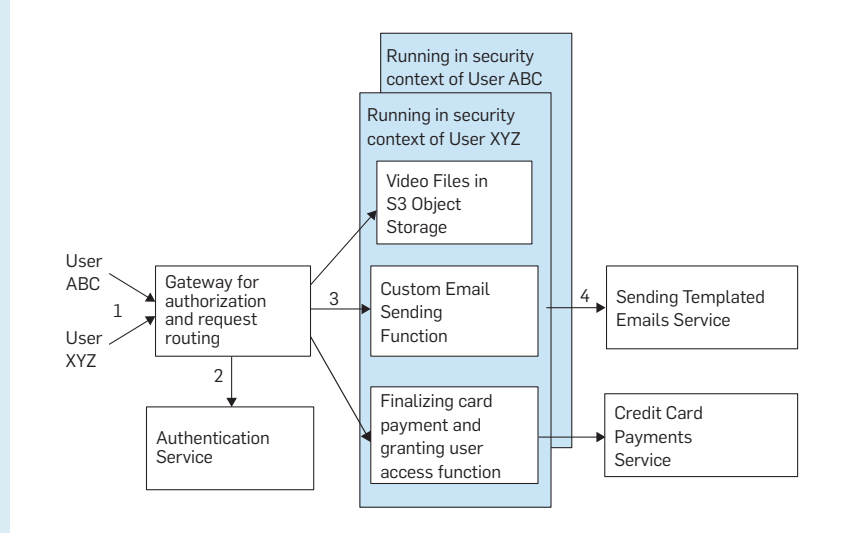
Programming models and tooling: since serverless functions are running for shorter amounts of time there will be multiple orders of magnitude more of them that compose applications (for example, SparqTV; <http://bit.ly/2xFktSb>), a video-streaming service runs more than 150 serverless functions). This however, will make it more difficult to debug and identify bottlenecks. Traditional tools that assumed access to servers (for example, root privilege) to monitor and debug applications are not applicable for serverless applications, and new approaches are needed. Although some of these tools are starting to become available, higher-level development IDEs, and tools for orchestrating and composing applications, will be critical. In addition, the platform may need to be extended with different recovery semantics, such as at-least-once or at-most-once, or more sophisticated concurrency semantics, such as atomicity where function executions are serialized. As well, refactoring functions (for example, splitting and merging them), and reverting to older versions, must be fully supported by the serverless platform. While these problems have received a lot of attention from industry and academia,²² there is still a lot of progress to be made.

Lack of standards and vendor lock-in: Serverless computing and FaaS are new and quickly changing and currently there are no standards. As the area matures, standards can

Use Case 4: Multi-Tenant Cloud Services

A Cloud Guru is a company whose mission is to provide users with cloud training that includes videos. An important part of their business model is providing service on-demand and optimizing delivery cost. Their usage patterns are unpredictable and may change depending on holidays or if they do promotions. They must be able to scale and to isolate users for security reasons while providing for each user backend functionality such as payment processing or sending email messages.

Requests are authenticated and routed to a custom function that runs in isolation and with the user's context.



They achieve this by leveraging cloud services and serverless computing to build a multi-tenant, secure, highly available, and scalable solution that can run each user specific code as serverless functions (<http://bit.ly/2JyPfbK>). This dramatically simplifies how a multi-tenant solution is architected as shown in the figure. A typical flow starts with a user making a request (1) from a frontend application (Web browser). The request is authenticated (2) by using an external service and then sent either to a cloud service (such as object store to provide video files) or (3) to a serverless function. The function makes necessary customizations and typically invokes other functions or (4) cloud services.

be expected to emerge. In the meantime, developers can use tools and frameworks that allow the use of different serverless computing providers interchangeably.

Research Opportunities

Since serverless is a new area, there are many opportunities for the research community to address. We highlight some options:

System-level research opportunities: A key differentiator of serverless is the ability to scale to zero, and not charge the customers for idle time. Scaling to zero, however, leads to problems of cold starts, particularly for functions with customized library requirements.¹⁷ Techniques to minimize the cold start problem while still scaling

to zero are critical. A more fundamental question currently being examined is if containers are the right abstractions for running serverless applications and whether abstractions with smaller footprints, such as unikernels, are more suitable.

Legacy code in serverless: Serverless application designs are fundamentally different from typical legacy applications. The economical value of existing code represents a huge investment of countless hours of developers coding and debugging software. One of the most important problems may be to what degree existing legacy code can be automatically or semiautomatically decomposed into smaller-granularity pieces to take advantage of these new economics.

Stateful serverless: Current serverless platforms are mostly stateless, and it is an open question if there will be inherently stateful serverless applications in the future with different degrees of QoS without sacrificing the scalability and fault-tolerance properties.

Service-level agreements (SLA): Serverless computing is poised to make developing services easier, but providing QoS guarantees remains difficult.^{17,27} While the serverless platform needs to offer some guarantees of scalability, performance, and availability, this is of little use if the application relies on an ecosystem of services, such as identity providers, messaging queues, and data persistence, which are outside the control of the serverless platform. To provide certain QoS guarantees, the serverless platform must communicate the required QoS requirements to the dependent components. Furthermore, enforcement may be needed across functions and APIs, through the careful measurement of such services, either through a third-party evaluation system, or self-reporting, to identify the bottlenecks.

Serverless at the edge: There is a natural connection between serverless functions and edge computing as events are typically generated at the edge with the increased adoption of IoT and other mobile devices. iRobot's use of AWS Lambda and step functions for image recognition was described by Barga as an example of an inherently distributed serverless application.⁸ Recently, Amazon extended its serverless capabilities to an edge based cloud environment by releasing AWS Greengrass. Consequently, the code running at the edge, and in the cloud may not just be embedded but virtualized to allow movement between devices and cloud. That may lead to specific requirements that redefine cost. For example, energy usage may be more important than speed.

New serverless applications: The serverless programming model is inherently different, but that should be a motivation to think about building—or rebuilding—new and innovative solutions that tap into what it can provide. Pywren,¹⁸ ExCamera,¹³ HPC, numerical analysis, and AI chatbots are but some

examples of how developers are using serverless to come up with new solutions and applications.

Conclusion

Serverless computing is an evolution in cloud application development, exemplified by the Function-as-a-Service model where users write small functions, which are then managed by the cloud platform. This model has proven useful in a number of application scenarios ranging from event handlers with bursty invocation patterns, to compute-intensive big data analytics. Serverless computing lowers the bar for developers by delegating to the platform provider much of the operational complexity of monitoring and scaling large-scale applications. However, the developer now needs to work around limitations on the stateless nature of their functions, and understand how to map their application's SLAs to those of the serverless platform and other dependent services. While many challenges remain, there have been rapid advances in the tools and programming models offered by industry, academia, and open source projects. ■

References

- Agha, G. An overview of actor languages. In *Proceedings of the 1986 SIGPLAN Workshop on Object-Oriented Programming*, 58–67. ACM, New York, NY.
- Armbrust, M. et al. A view of cloud computing. *Commun. ACM* 53, 4 (2010), 50–58; <https://m.cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext>
- AWS re:invent 2014—(mb1202) new launch: Getting started with AWS lambda; <https://www.youtube.com/watch?v=UFJ27laTWQA>.
- Bainomugisha, E., Carreton, A.L., Cutsem, V., Mostinckx, S. and Meuter, W.D. A survey on reactive programming. *ACM Comput. Surv.* 45, 4 (Aug. 2013), 52:1–52:34.
- Baldini, I., Castro, P., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Suter, P. Cloud-native, event-based programming for mobile applications. In *Proceedings of the Intern. Conf. on Mobile Software Engineering and Systems*, 2016, 287–288. ACM, New York, NY.
- Baldini, I. et al. Serverless computing: Current trends and open problems. *Research Advances in Cloud Computing*, Springer, 2017, 1–20.
- Baldini, I., Cheng, P., Fink, S.J., Mitchell, N., Muthusamy, V., Rabbah, R., Suter, P. and Tardieu, O. The serverless trilemma: Function composition for serverless computing. In *Proceedings of the 2017 ACM SIGPLAN Intern. Symp. on New Ideas, New Paradigms, and Reflections on Programming and Software*.
- Barga, R.S. Serverless computing: Redefining the cloud [Internet]. In *Proceedings of the 1st Intern. Workshop on Serverless Computing* (Atlanta, GA, USA, June 5, 2017); <http://www.serverlesscomputing.org/wosc17/#keynote>
- Bernstein, D. Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing* 1, 3 (Sept. 2014), 81–84.
- Businesswire. \$7.72 billion function-as-a-service market 2017—Global forecast to 2021: Increasing shift from DevOps to serverless computing to drive the overall Function-as-a-Service market; <https://bwnnews.pr/2G3ZzQY>.
- CNCF Serverless White Paper; <https://github.com/>

- cncc/wg-serverless#whitepaper
- Etzioni, O. and Niblett, P. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, 2010.
- Fouladi, S., Wahby, R.S., Shacklett, B., Balasubramaniam, K., Zeng, W., Bhalerao, R., Sivaraman, A., Porter, G. and Winstead, K. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. *NSDI* (2017), 363–376.
- Fox, G.C., Ishakian, V., Muthusamy, V. and Slominski, A. Status of Serverless Computing and Function-as-a-Service (FaaS) in Industry and Research. Technical Report; arXiv:1708.08028, 2017.
- Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H. Serverless computation with openlambda. In *Proceeding of the 8th USENIX Workshop on Hot Topics in Cloud Computing* (Denver, CO, USA, June 20–21, 2016).
- IDC. IDC FutureScape: Worldwide IT Industry 2017 Predictions. IDC #US41883016, 2016.
- Ishakian, V., Muthusamy, V. and Slominski, A. Serving deep learning models in a serverless platform. In *Proceedings of the IEEE Intern. Conf. on Cloud Engineering*, 2018.
- Jonas, E., Pu, Q., Venkataraman, S., Stoica, I. and Recht, B. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the 2017 Symp. on Cloud Computing*.
- Kilcioglu, C., Rao, J.M., Kannan, A. and McAfee, R.P. Usage patterns and the economics of the public cloud. In *Proceedings of the 26th Intern. Conf. World Wide Web*, 2017.
- Lee, H., Satyam, K. and Fox, G.C. Evaluation of production serverless computing environments. In *Proceedings of IEEE Cloud Conf. Workshop on Serverless Computing* (San Francisco, CA, 2018).
- Leitner, P., Wittern, E., Spillner, J. and Hummer, W. A mixed-method empirical study of Function-as-a-Service software development in industrial practice; <https://peerj.com/preprints/27005>
- Lin, W.-T., Krintz, C., Wolski, R., Zhang, M., Cai, X., Li, T. and Xu, W. Tracking causal order in AWS lambda applications. In *Proceedings of the IEEE Intern. Conf. on Cloud Engineering*, 2018.
- NGINX. NGINX announces results of 2016 future of application development and delivery survey; <http://bit.ly/2YM27e2/>.
- Oakes, E., Yang, L., Houck, K., Harter, T., Arpaci-Dusseau, A.C. and Arpaci-Dusseau, R.H. Pipsqueak: Lean Lambdas with large libraries. In *Proceedings of 2017 IEEE 37th Intern. Conf. on Distributed Computing Systems Workshops*, 395–400.
- Paton, N.W. and Díaz, O. Active database systems. *ACM Comput. Surv.* 31, 1 (1999), 63–103.
- Wang, L., Li, M., Zhang, Y., Ristenpart, T. and Swift, M. Peeking behind the curtains of serverless platforms. In *Proceedings of USENIX Annual Technical Conf.*, 2018, 133–146. USENIX Association.
- Yan, M., Castro, P., Cheng, P., Ishakian, V. Building a chatbot with serverless computing. In *Proceedings of the 1st Intern. Workshop on Mashups of Things*, 2016.
- Ye, W., Khan, A.I. and Kendall, E.A. Distributed network file storage for a serverless (P2P) network. In *Proceedings of the 11th IEEE Intern. Conf. on Networks*, 2003, 343–347.

Paul Castro (castrop@us.ibm.com) is a research staff member at IBM T.J. Watson Research Center in Yorktown Heights, NY, USA.

Vatche Ishakian (vishakian@bentley.edu) is an assistant professor at Bentley University, Waltham, MA, USA.

Vinod Muthusamy (vmuthus@us.ibm.com) is a research scientist at IBM Research AI in Austin, TX, USA.

Aleksander Slominski (<https://aslom.net>) is research staff member in the Serverless Group in Cloud Platform, Cognitive Systems and Services Department at IBM T.J. Watson Research Center in Yorktown Heights, NY, USA.

© 2019 ACM 0001-0782/19/12 \$15.00



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/the-rise-of-serverless-computing>

Today's Research Driving Tomorrow's Technology

The ACM Digital Library (DL) is the most comprehensive research platform available for computing and information technology and includes the ongoing contributions of the field's most renowned researchers and practitioners.

Each year, roughly 20,000 newly published articles from ACM journals, magazines, technical newsletters and annual conference volumes are added to the DL's complete full text contents of more than 550,000 articles.

The DL also features the fully integrated and comprehensive bibliographic index, *The Guide to Computing Literature*—a continually updated index featuring millions of publication records from over 5,000 publishers worldwide.

For more information, please visit

<https://libraries.acm.org/>

or contact ACM at

dl-info@hq.acm.org

ACM

DL

DIGITAL
LIBRARY

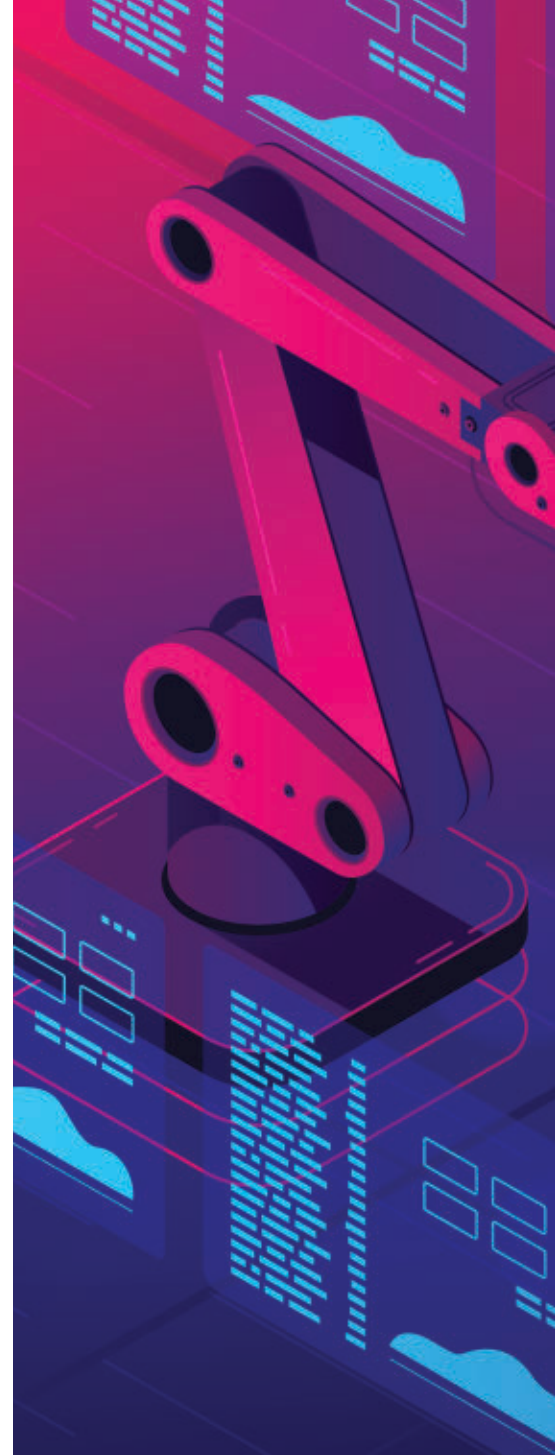
Automated program repair can relieve programmers from the burden of manually fixing the ever-increasing number of programming mistakes.

BY CLAIRE LE GOUES, MICHAEL PRADEL, AND ABHIK ROYCHOUDHURY

Automated Program Repair

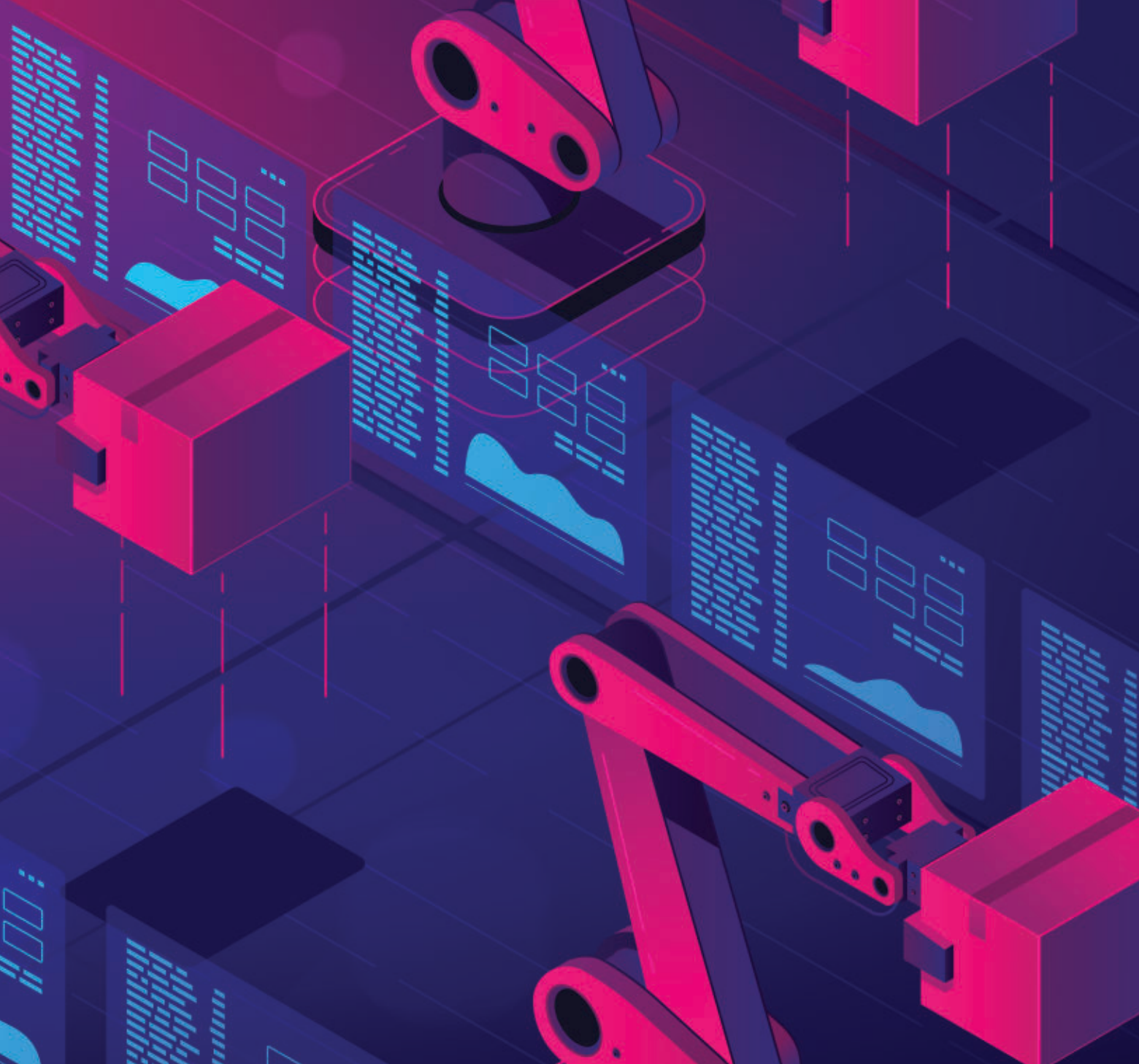
ALEX IS A software developer, a recent hire at the company of her dreams. She is finally ready to push a highly anticipated new feature to the shared code repository, an important milestone in her career as a developer. As is increasingly common in development practice, this kind of push triggers myriads of tests the code must pass before becoming visible to everyone in the company. Alex has heavily tested the new feature and is confident it will pass all the tests automatically triggered by the push. Unfortunately, Alex learns the build system rejected the commit. The continuous integration system reports failed tests associated with a software package developed by a different team entirely. Alex now must understand the problem and fix the feature manually.

What if, instead of simply informing Alex of the failing test, the build system also suggested one or two possible patches for the committed code? Although this general use case is still fictional, a growing community



» key insights

- Automated program repair is an emerging and exciting field of research that allows for automated rectification of software errors and vulnerabilities.
- The uses of automated program repair can be myriad, such as improving programmer productivity, automated fixing of security vulnerabilities, self-healing software for autonomous devices, and automatically generating hints for solving programming assignments.
- Automated repair can benefit from various techniques: intelligent navigation over a search space of program edits, symbolic reasoning to synthesize suitable code fragments, and techniques that learn from existing code and patches.



of researchers is working on new techniques for automated program repair that could make it a reality. A bibliography of automated program repair research has been composed by Monperrus.¹⁷

In essence, automated repair techniques try to automatically identify patches for a given bug,^a which can then be applied with little, or possibly even without, human intervention. This type of work is beginning to see adoption in certain, constrained, prac-

tical domains. Static bug finding tools increasingly provide “quick fix” suggestions to help developers address flagged bugs or bad code patterns, and Facebook recently announced a tool that automatically suggests fixes for bugs found via their automatic testing tool for Android applications.¹⁵

The problem of bugs motivates a broad array of work on automatically identifying them. Advances in formal verification have shown the promise of fully assured software. However, the pace and scale of modern software development often precludes the application of such techniques from all but

the most safety-critical systems. Lightweight static approaches that rely most commonly on syntactic pattern matching or less complex static analysis are becoming increasingly popular as quality gates in many companies.^{7,23} Testing, at multiple levels of system abstraction, remains the most common bug detection technique in practice.

While detecting bugs is a necessary step toward improving software, it leaves the arguably harder task of fixing bugs unsolved. In practice, program repair is challenging for several reasons. A developer must at first understand the problem and localize its root cause in the source code.

^a We use the colloquial term “bug” to refer to programming mistakes that result in unintended runtime behavior.

Next, she must speculate about strategies to possibly fix the problem. For some of these strategies, the developer will evaluate a potential patch, by applying it and evaluating whether the associated test cases then pass; if not, she might use the failing test cases to conduct additional debugging activities. Finally, the developer must select a patch and apply it to code base. The difficulty of all these tasks is compounded by the fact that complex software projects tend to contain legacy code, code written by other members of an organization, or even code written by third parties.

The promise of automated program repair is in reducing the burden of these tasks by suggesting likely correct patches for software bugs. At a high level, such techniques take as input a program and some specification of the correctness criteria that the fixed program should meet. Most research techniques use test suites for this purpose: one or more failing tests indicate a bug to be fixed, while passing tests indicate behavior that should not change. The end goal is a set of program changes (typically to source code) that leads all tests to pass, fixing the bug without breaking other behavior.

The grand challenge in today's research on automated program repair is the problem of weak specifications. Since detailed formal specifications of intended program behavior are typically unavailable, program repair is driven by weak correctness criteria, such as a test suite. As a result, the generated patches may over-fit the given test suite and may not generalize to tests outside the test suite.²⁹

In the rest of this article, we discuss some of the technical developments in automated program repair, including an illustration of the overfitting problem. We start by sketching some of the use-cases of automated program repair.

Use Cases

This section discusses four practical use cases of automated repair, and reports initial experience based on current repair techniques.

Fixing bugs throughout development. Existing continuous integration (CI) pipelines, such as Jenkins, are an important stepping stone for integrating repair into the development process. By regularly building, testing, and



The grand challenge in today's research on automated program repair is the problem of weak specifications.



deploying a code base, CI provides the prerequisites for repair tools that use test suites as correctness specifications. Repair can become an activity in CI systems that suggests patches in response to regression test failures, such as for Alex, our hypothetical programmer.

Are we there yet? Existing techniques for automated repair of correctness bugs are typically evaluated for effectiveness using bugs taken from open source projects. Because many techniques require input tests to trigger the bug under repair and to evaluate the technique, such programs and bugs must be associated with one or more failing test cases. These bugs are typically collected systematically by going back in time through code histories to identify bug-fixing commits and the regression tests associated with them. Open source projects whose bugs have been studied in this way include popular Java projects, for example, various Apache libraries, Log4J, and the Rhino JavaScript interpreter, as well as popular C projects, for example, the PHP and Python interpreters, the Wireshark network protocol analyzer, and the libtiff library.

Recently, the Repairnator project³³ has presented a bot which monitors for software errors, and automatically find fixes using repair tools. Another recent work from Facebook¹⁵ describes experiences in integrating repair as part of continuous integration—a repair tool monitors test failures, reproduces them, and automatically looks for patches. Once patches are found, they are presented to the developers for validation. Currently, the effort focuses on automatically repairing crashes in Android apps, however, the project plan is to extend the work to general-purpose repair.

Repairing security vulnerabilities.

Many security vulnerabilities are exploitable memory errors or programming errors, and hence a relevant target for automated repair. Key software, including popular libraries processing file formats or operating system utilities, are regularly and rigorously checked for vulnerabilities in response to frequent updates using grey-box fuzz testing tools, such as American Fuzzy Lop (AFL^b). Microsoft recently

^b <http://lcamtuf.coredump.cx/afl/>

Figure 1. Simple example for categorizing triangles.

```

1 int triangle(int a, int b, int c){
2     if (a <= "0" || b <= "0" || c <= "0")
3         return INVALID;
4     if (a == b && b == c)
5         return EQUILATERAL;
6     if (a == b || b != c)    // bug!
7         return ISOSCELES;
8     return SCALENE;
9 }

```

announced the Springfield project; Google similarly announced the OSS-Fuzz project. Such continuous fuzzing workflows generate use cases for automated program repair. In particular, repair tools can receive tests produced by grey-box fuzz testing tools like AFL.

Are we there yet? Existing repair techniques are effective at fixing certain classes of security vulnerabilities, specifically integer and buffer overflows. An empirical study conducted on OSS Fuzz subjects^c shows that integer overflow errors are amenable to one-line patches, which are easily produced by repair tools. For example, these changes add explicit casts of variables or constants, modify conditional checks to prevent overflows, or change type declarations. Existing repair tools¹⁶ have also been shown to automatically produce patches for the infamous Heartbleed vulnerability:

```

if (hbtype == TLS1_HB_REQUEST
    /* the following check being added
    is the fix */
    && payload + 18 < s->s3->rrec.
length) {
    ...
    memcpy(bp, pl, payload);
    ...
}

```

It is functionally equivalent to the developer-provided patch:

```

/* the following check being added is
the fix */
if (1 + 2 + payload + 16 > s->s3->rrec.
length) return 0;
...
if (hbtype == TLS1_HB_REQUEST) {
    ...
}

```

c <https://github.com/google/oss-fuzz>

Intelligent tutoring. The computer programming learning community is growing rapidly.

This growth has increasingly led to large groups of potential learners, with often inadequate teaching support. Automated repair can serve as a key component of intelligent tutoring systems that provide hints to learners for solving programming assignments and that automate the grading of students' programming assignments by comparing them with a model solution.

Are we there yet? While repair-based intelligent tutoring remains an open challenge for now, initial evidence on using program repair like processes for providing feedback to students²⁸ or for automatic grading of student assignments⁴⁰ have been obtained. Automated assignment grading can benefit from computation of the "semantic distance" between a student's buggy solution and an instructor's model solution. An important challenge for the future is that programming education requires nuanced changes to today's program repair workflow, since teaching is primarily focused on guiding the students to a solution, rather than repairing their broken solution.

Self-healing of performance bottlenecks. With the emergence of a wide variety of Internet of Things (IoT) software for smart devices, drones, and other cyber-physical or autonomous systems, there is an increasing need for online program repair, especially for non-functional properties like energy consumption. Consider a drone used for disaster recovery, such as flood or fire control. The drone software may encounter unexpected or perilous inputs simply by virtue of being faced with an unforeseen physical environment, which may drain the device's battery. There exists a need for online self-healing of the drone software. Au-

tomated repair targeted at non-functional issues, such as performance bottlenecks, can provide such self-healing abilities.

Are we there yet? Current repair techniques for non-functional properties have shown their effectiveness in improving real-world software. Consider two examples of performance-related repair tools. First, the MemoizeIt tool³¹ suggests code that performs application-level caching, which allows programs to avoid unnecessarily repeated computations. Second, the Caramel tool¹⁹ has suggested patches for a total of 150 previously unknown performance issues in widely used Java and C/C++ programs, such as Lucene, Chromium, and MySQL, that are now fixed based on the suggested repairs. While these examples are encouraging, the question of how to apply non-functional repair for fully automated self-healing remains open.

Simple Example

Here, we describe a simple example we will use to illustrate the various state-of-the-art techniques in program repair. The example is selected for didactic purposes rather than to illustrate all the capabilities of repair techniques. Today's techniques apply to significantly more complex programs, as we described previously.

Consider a function that categorizes a given triangle as scalene, isosceles, or equilateral (Figure 1). From the definition of isosceles triangles learned in middle school, we can see that the condition in line 6 should be rectified to

```
(a == b || b == c || a == c)
```

This repair is non-trivial; it goes beyond simply mutating one operator in the condition.

The test suite in Figure 2 captures

Figure 2. Test suite for the function in Figure 1.

Test-id	a	b	c	Expected output	Pass/Fail
1	-1	-1	-1	INVALID	Pass
2	1	1	1	EQUILATERAL	Pass
3	2	2	3	ISOSCELES	Pass
4	3	2	2	ISOSCELES	Fail
5	2	3	2	ISOSCELES	Fail
6	2	3	4	SCALENE	Fail

the various triangle categories considered by the function: INVALID, EQUILATERAL, ISOSCELES, and SCALENE. Because the code contains a bug, several of the tests fail. The goal of automated program repair is to take a buggy program and a test suite, such as these, and produce a patch that fixes the program. The test suite provides the correctness criterion in this case, guiding the repair toward a valid patch. In general, there may exist any number of patches for any particular bug, and even humans can find different patches for real-world bugs.

At a high level, the program repair problem can be seen as follows: *Given a buggy program P , and a test suite T such that P fails one or more tests in T , find a “small” modification of P such that the modified program P' passes T .* The term “small” simply refers to the fact that developers generally prefer a simpler patch over a complicated one. Some techniques even try to find a minimal patch. Others trade off patch size with other goals, such as finding a patch efficiently. A particular risk in automated repair is a “patch” that causes the provided test cases to pass but that does not generalize to the complete, typically unavailable, specification. That is, the patch produced by an automated repair method can overfit the test data. An extreme case of an overfitted repair for the tests in Figure 2 is the following:

```
if (a==-1 && b==-1 && c==-1)
    return INVALID;
if (a==1 && b==1 && c==1)
```

```
    return EQUILATERAL;
if (a==2 && b==2 && c==3)
    return ISOSCELES;
...
```

Of course, such a “repaired” program is not useful since it does not pass any tests outside the provided test suite. This example is deliberately extreme. More commonly, patches produced by current repair techniques tend to overfit the provided test suite by disabling (or deleting) undertested functionality.²⁹

State of the Art

Automatically repairing a bug involves (implicitly) searching over a space of changes to the input source code. Techniques for constructing such patches can be divided into broad categories, based on what types of patches are constructed, and how the search is conducted. Figure 3 gives an overview of the techniques. The inputs to these techniques are a buggy program and a correctness criterion (the correctness criterion is often given as a test suite). Most techniques start with a common preprocessing step that identifies those code locations that are likely to be buggy. Such a fault localization procedure, for example, that by Jones et al.,⁸ provides a ranking of code locations that indicates their potential buggy-ness. At a high level, there are two main approaches: *heuristic repair* and *constraint-based repair*. These techniques can sometimes be enhanced by machine learning, which we call *learning-aided repair*.

Heuristic repair. Heuristic search methods, shown at the left of Figure 3 employ a generate-and-test methodology, constructing and iterating over a search space of syntactic program modifications. These techniques can be explained schematically as follows:

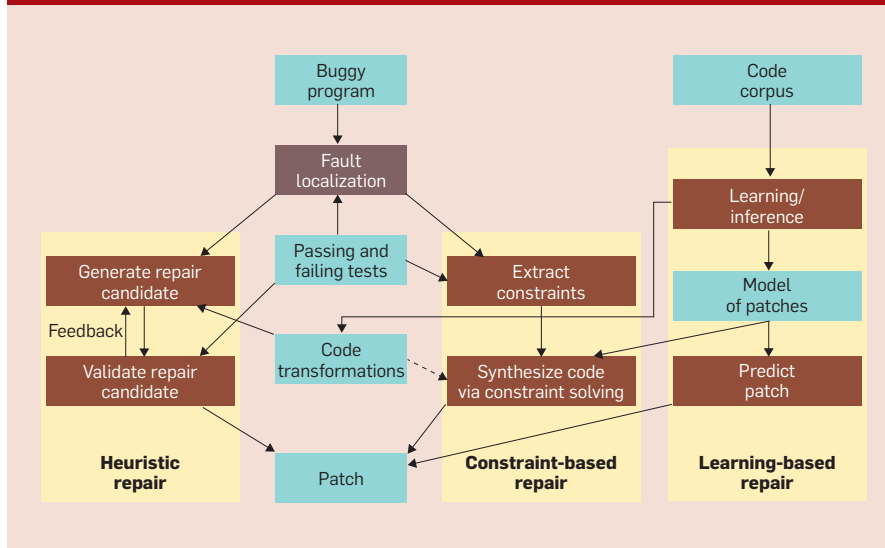
```
for cand ∈ SearchSpace do
    validate cand // break if successful
done
```

with *SearchSpace* denoting the set of considered modifications of the program. Validation involves calculating the number of tests that pass when a suggested patch has been applied. This can amount to a fitness function evaluation in genetic programming or other stochastic search methods.

Heuristic repair operates by generating patches that transform the program abstract syntax tree (AST). An AST is a tree-based representation of program source code that captures important program constructs, while abstracting away syntactic details like parentheses or semicolons. Given fault localization information that pinpoints code locations in the program that are the most likely to be buggy, syntactic techniques render the search tractable by making choices along one of three axes: mutation selection, test execution, and the traversal strategy.

Mutation selection. Due to the combinatorial explosion of possible mutations, the number of program variants that can be generated and compiled is typically very large. Techniques thus must limit the type and variety of edits considered for a repair candidate. This in turn defines the search space, with which search-based repair algorithms have great flexibility. However, this flexibility comes at a risk: If the search space is too small, the desired repair may not even be in the search space. For our triangle example (Figure 1), recall that the most natural patch replaces line 6 with $(a == b \parallel b == c \parallel a == c)$. If we only consider mutations that modify binary operators, the single-edit search space of the repair algorithm will not contain the developer-provided repair, which requires augmenting the branch condition with new conditions. On the other hand, if the search space is too large, the search can become intrac-

Figure 3. Overview of repair techniques.



table, such that the repair may not be found by the algorithm in a reasonable amount of time.

To address this issue, some techniques limit edits to only deletion, insertion, or replacement of code at the statement- or block-level. For code insertion or replacement, a common approach is to pull code from elsewhere in the same program or module, following the plastic surgery hypothesis (that correct code can be imported from elsewhere in the same program)⁶ or the competent programmer hypothesis (that programmers are mostly competent, and while they may make a mistake in one portion of a program, they are likely to have programmed similar logic correctly, elsewhere). Such a technique would therefore only consider moving entire blocks or lines of code around, for example, an entire if condition semantically similar to the one shown in Figure 1. This can often work by virtue of the fact that source code is repetitive.²²

Other techniques have benefited from using more expressive transformation templates, such as guarding a de-reference operation with a null-pointer check. Such transformation templates trade off repair space size for readability and “naturalness” of the resulting patches. Moving from statement-level edits to expression-level edits increases the search space, with the amount of increase depending on the transformation templates used to construct the search space.

However, even if the search space is large, the mutation operators may not support the behavioral change needed by the program or may affect the desired change in ways different from what a human might propose. A technique that may modify operators or insert conditions (copied from elsewhere in the program) would still struggle on this small program, since `(a == c)` never appears verbatim in our example. Such a lack of correct code fragments can result in degenerate behavior on smaller programs that provide little repair material. It also motivates research in intelligently augmenting the search space, for example, by considering past versions of a program.

Test execution. Repair candidates are evaluated by running the modified program on the provided set of test cases. Test execution is typically the most

expensive step, as test suites can be large and techniques may need to re-run them many times. Various strategies have been proposed to reduce this cost, including test suite selection and prioritization. Search strategies that do not require a fitness function, for example, based on random or deterministic search, can reduce the cost of testing by simply failing early (at the first test failure). Moreover, such techniques may run the test cases in a heuristic order designed to maximize the chance that, if a test case is going to fail, it is run early in the validation process.

Traversal strategy. Finally, techniques vary in how they choose which patches to explore, and in what order. GenProg,³⁷ an early technique proposed in this domain, implements a genetic programming algorithm with a fitness function based on the number of test cases passed by a patched program. Subsequent techniques like PAR⁹ have followed, varying in the mutation operators (PAR) or the fitness function. Other techniques simply sample randomly typically restricting themselves to single-edit patches,²¹ or in a heuristic, deterministic order as in GenProg AE.³⁶

Constraint-based repair. In contrast to heuristic repair techniques, constraint-based techniques proceed by constructing a repair constraint that the patched code should satisfy, as illustrated in Figure 3. The patch code to be generated is treated as a function to be synthesized. Symbolic execution or other approaches extract properties about the function to be synthesized; these properties constitute the repair constraint. Solutions to the repair constraint can be obtained by constraint solving or other search techniques. In these approaches, formulation of the repair constraint is the key, not the mechanism for solving it. This class of techniques can be captured via the following schematic:

for test $t \in \text{test-suite } T$
 compute repair constraint ψ_t
 synthesize e as solution for $\forall_t \psi_t$

In this case, T is the test suite used as the correctness criterion to guide repair. The constraint ψ_t will be computed via a symbolic execution¹⁰ of the

path traversed by test $t \in T$. The constraint ψ_t is often of the form

$$\psi_t \equiv \pi_t \wedge \text{output} = \text{expected}$$

where π_t is the path condition of the path traversed by test t , output is the symbolic expression capturing the output variable in the execution of t and expected captures the oracle or expectation. The path condition of a program path is a formula, which is true for those inputs which traverse the path.¹⁰

Computing repair constraints and angelix values. To illustrate constraint-based repair, reconsider our running example from earlier. SemFix,¹⁸ which is representative for constraint-based techniques, substitutes the faulty condition in line 6 with an abstract function $f(a,b,c)$ on the live variables. In this example, f is a predicate that takes the integer values a,b,c and returns true/false. Then, the technique symbolically executes the tests in the given test suite to infer properties about the unknown function f . These properties are crucial for synthesizing an appropriate f that can pass all the given test cases.

The first two tests in our test suite do not even reach line 6. Hence, SemFix will not infer any property about the function f from them. From the last four tests, it can infer the repair constraint

$$f(2,2,3) \wedge f(3,2,2) \wedge f(2,3,2) \wedge f(2,3,4)$$

This is because analysis of the program has revealed that for input exercising line 6 if f is true, the program returns ISOSCELES and otherwise SCALENE.

Inferring detailed constraint specifications can be difficult, sometimes posing significant scalability issues. This motivates more efficient inference of value-based specifications.¹⁶ In particular, *angelic values* are inferred for patch locations, where an angelic value for a fixed location is one that makes a failing test case pass. Once (collections of) angelic values are identified for each failing test, program synthesis can then be employed to generate patch code meeting such a value-based specification. This is the philosophy embodied in the Angelix tool¹⁶ where angelic values are obtained via symbolic execution (instead of producing repair specifications in the form of SMT constraints via symbolic execution directly). This way of dividing the

repair task into angelic value determination and patch code generation to meet angelic values is symptomatic of semantic repair approaches.

Instead of obtaining angelic values by symbolic execution and constraint solving, they may also be obtained by search, particularly for conditional statements. This is because each occurrence of a conditional statement has only two possible return values: true and false. Techniques that work on enumerating possible angelic values without adopting symbolic execution^{13,39} typically try to repair conditional statements exclusively, where the angelic values are exhaustively enumerated until all failing test cases pass. Such techniques adopt the work-flow of semantic repair techniques (specification inference followed by patch generation), with an enumeration step fully or partially replacing symbolic program analysis. Symbolic analysis-based approaches such as Mechtaev et al.¹⁶ on the other hand, avoid exhaustive enumeration of possible angelic values.

Solving constraints to find a patch. Once repair constraints or angelic value(s) of a statement to be fixed are obtained, these techniques must generate a patch to realize the angelic value. Finding a solution to the repair constraint yields a definition of the abstract function f , which corresponds to the patched code. This is often achieved by either search or constraint solving, where the operators allowed to appear in the yet-to-be synthesized function f are restricted. In this example, if we restrict the operators allowed to appear in f to be relational operators most search or solving techniques will find the expression $a == b \ || \ b == c \ || \ a == c$. Efficient program synthesis techniques (see Alur et al.² for an exposition of some recent advances in program synthesis) are often used to construct the function f .

Learning-based repair. Recent improvements in advanced machine learning, especially deep learning, and the availability of large numbers of patches enable learning-based repair. Current approaches fall approximately into three categories that vary by the extent to which they exploit learning during the repair process. One line of work¹⁴ learns from a corpus of code a model of correct code, which indicates how likely a given piece of code is with regard to the code

corpus. The approach then uses this model to rank a set of candidate patches to suggest the most realistic patches first. Another line of work infers code transformation templates from successful patches in commit histories.^{3,12} In particular, Long et al.¹² infers AST-to-AST transformation templates that summarize how patches modify buggy code into correct code. These transformation templates can then be used to generate repair candidates.

The third line of work not only improves some part of the repair process through learning, but also trains models for end-to-end repair. Such a model predicts the repaired code for a given piece of buggy code, without relying on any other explicitly provided information. In particular, in contrast to the repair techniques discussed previously, such models do not rely on a test suite or a constraint solver. DeepFix⁵ trains a neural network that fixes compilation errors, for example, missing closing braces, incompatible operators, or missing declarations. The approach uses a compiler as an oracle to validate patch candidates before suggesting them to the user. Tufano et al.³² propose a model that predicts arbitrary fixes and trains this model with bug fixes extracted from version histories. According to their initial results, the model produces bug-fixing patches for real defects in 9% of the cases. Both approaches abstract the code before feeding it into the neural network. For the running example in Figure 1, this abstraction would replace the application-specific identifiers `triangle` and `EQUILATERAL` with generic placeholders, such as `VAR1` and `VAR2`. After this abstraction, both approaches use an RNN-based sequence-to-sequence network that predicts how to modify the abstracted code.

Given the increasing interest in learning-based approaches toward software engineering problems, we will likely see more progress on learning-based repair in the coming years. Key challenges toward effective solutions include finding an appropriate representation of source code changes and obtaining large amounts of high-quality human patches as training data.

Repair of non-functional properties. To help developers improve the

software efficiency, several approaches identify optimization opportunities and make suggestions on how to refactor the code to improve performance. These approaches typically focus on a particular kind of performance problem, for example, unnecessary loop executions¹⁹ or repeated executions of the same computation.³¹ Another line of work selects which data structure is most likely to provide the best performance for a given program out of a given set of functionally equivalent data structures.²⁶ All these approaches suggest code changes but leave the final decision whether to apply an optimization to the developer.

To mitigate security threats, various techniques for repairing programs at runtime have been proposed. These approaches automatically rewrite code to add a runtime mechanism that enforces some kind of security policy. For example, such repair techniques can enforce control flow integrity,¹ prevent code injections,³⁰ automatically insert sanitizers of untrusted input, or enforce automatically inferred safety properties.²⁰

We note that existing techniques to repair non-functional properties typically focus on a particular kind of problem, for example, a kind of performance anti-pattern or attack. This distinguishes them from the core repair literature for fixing correctness bugs, which typically aim at fixing a larger set of errors.

Perspectives and Challenges

Despite tremendous advances in program repair during the last decade, there remain various open challenges to be tackled by future work. We identify three core challenges: increasing and ensuring the quality of repairs; extending the scope of problems addressed by repair; and integrating repair into the development process.

Quality. The quality challenge is about increasing the chance an automatically identified repair provides a correct fix that is easy to maintain in the long term. Addressing this challenge is perhaps the most important step toward real-life adoption of program repair.

Measures of correctness. An important aspect of fix quality is whether the fix actually corrects the bug. In practice, program repair relies on measures

of correctness. Finding such a measure is a difficult and unsolved problem, which applies both to patches produced by humans and by machines. To date, researchers have assessed quality using human judgment, crowdsourced evaluations, comparison to developer patches of historical bugs, or patched program performance on indicative program workloads or held-out test cases. The recent work of Xiong et al.³⁸ provides a novel outlook for filtering patches based on the behavior of the patched program vis-a-vis the original program on passing and failing tests.

Alternative oracles. The bulk of the existing literature focuses on test-based repair where the correctness criteria is given as a test suite. Richer correctness properties, for example, assertions or contracts, can be used to guide repair when available.³⁴ Other approaches consider alternative oracles, such as potential invariants inferred from dynamic executions.²⁰ Such approaches can follow the “bugs as deviant behavior” philosophy, where deviations of an execution from “normal” executions are observed and avoided. In particular, Weimer et al.³⁵ provide an overview of various (partial) oracles that can be used for repair.

Correctness guarantees. Few of today’s repair techniques provide any guarantees about the correctness of produced patches, which can hinder the application of automated repair, especially to safety-critical software. If correctness guarantees are available as properties, such as pre-conditions, post-conditions, and object invariants, these can be used to guide program repair. The work of Logozzo and Ball¹¹ reports such an effort where repair attempts to increase the number of property-preserving executions, while reducing the number of violating executions. However, such formal techniques are contingent on the properties to drive the repair being available.

Maintainability. Once a correct fix has been detected and applied to the code base, the fixed code should be as easy to maintain as a human fix. Initial work in this domain has investigated the effect that automatically generated patches impact human maintenance behavior.⁴ More study is needed to develop a foundational understanding of change quality, especially with respect



Once repair constraints or angelic value(s) of a statement to be fixed are obtained, these techniques generate a patch to realize the angelic value.



to the human developers who will interact with a modified system.

A promising avenue for tackling the quality challenge is by leveraging information available from other development artifacts, including documentation or formal specifications, language specifications and type systems, or source control histories of either the program under repair or of the broad corpora of freely available open source software. Such additional information can reduce the repair search space by imposing new constraints on potential program modifications (for example, as suggested by a type system) and increase the probability that the produced patch is human-acceptable.

Scope. The scope challenge is about further extending the kinds of bugs and programs to which automated repair applies.

General-purpose repair. Research in program analysis has long focused on special-purpose repair tools for specific kind of errors, such as buffer overflow errors,²⁰ or bugs in domain-specific languages.²⁴ More recent work, as discussed earlier, focuses on general-purpose repair tools that do not make any assumptions about the kind of bugs under consideration. While automatically fixing all bugs seems out of reach in the foreseeable future, targeting a broad set of bugs remains an important challenge.

Complex programs and patches. Many of the key innovations in the initial research in program repair concerned the scalability of techniques to complex programs. For example, search-based techniques moved from reasoning over populations of program ASTs to populations of small edit programs (the patches themselves) and developed other techniques to effectively constrain the search space. Constraint-based repair strategies have moved from reasoning about the semantics of entire methods to only reasoning about the desired change in behavior. These efforts enable scaling to programs of significant size, and multi-line repairs.¹⁶ We anticipate that scalability will periodically return to the fore as program repair techniques engage in more complex reasoning. We emphasize here that program repair techniques should remain scalable with respect to large

programs as well as large search spaces (complex changes).

Development process. The final challenge is about integrating repair tools into the development process.

Integration with bug detection. Bug detection is the natural step preceding program repair. It is possible to fuse debugging and repair into one step, by viewing repair as the minimal change, which makes the program pass the given tests. We envision future work integrating repair with bug detection techniques, such as static analysis tools. Doing so may enable repair techniques to obtain additional information about possible repairs from static analyses, in addition to the test cases used nowadays. As a first step in this direction, a static analysis infrastructure used at Google suggests fixes for a subset of its warnings.²³ A promising future direction here could be to extend static analysis tools for generating dynamic witnesses or scenarios of undesirable behavior.

IDE integration. Most of today's repair tools are research prototypes. Bringing these tools to the fingertips of developers in a user-friendly fashion will require efforts toward integrating repair into integrated development environments (IDEs). For example, an IDE-integrated repair tool could respond to either failed unit or system tests or developer prompting. To the best of our knowledge, this application has not yet been widely explored. Suitably interactive response times are a precondition for such an approach. This research direction will benefit from interaction with experts in developer tooling and human-computer interaction, to ensure tools are designed and evaluated effectively.

Interactivity. As program repair gets integrated into development environments, interacting with the developer during repair is important. While the focus in the past decade has been on fully automated repair, putting the developer back into the loop is necessary, in particular, due to the weak specifications (test suites) often used to guide program repair. User interactivity may be needed to yield expected outputs of additional test inputs that are generated to strengthen the test-suite driving repair.²⁷ It is of course possible to filter plausible



Automated program repair remains an enticing yet achievable possibility that can improve program quality while improving programmers' development experience.



but possibly incorrect patches by, for example, favoring smaller patches or favoring patches “similar” to human patches. Nevertheless, the developer still needs to explore the remaining large set of patch suggestions.

Explaining repairs. A strongly related problem is to explain repair suggestions. One idea worth pursuing is to compute and present the correlation of patches based on program dependencies and other semantic features, which allow the developer to loosely group together plausible patches. Explaining repairs is needed particularly in its application to programming education.⁴⁰ Instead of merely fixing a students' incorrect program to the model correct program, it is useful for the repair tool to generate hints of what is missing in the students' repair. Such hints may take the form of logical formulae capturing the effect of repairs, which are gleaned from constraint-based repair tools; these hints may be presented in natural language, instead of logic, for easy comprehension by the learners.

Conclusion

Automated program repair remains an enticing yet achievable possibility that can improve program quality while improving programmers' development experience.

Technically speaking, automated repair involves challenges in defining, navigating, and prioritizing the space of patches. The field benefits from past lessons learned in search space definition and navigation in software testing, as embodied by the vast literature in test selection and prioritization. The GenProg tool³⁷ is just one example of how genetic search, which has been useful for testing, can be potentially adapted for repair. At the same time, automated repair comes with new challenges because it may generate patches that overfit the given tests. This is a manifestation of tests being incomplete correctness specifications. Thus, there is a need for inferring specifications to guide repair, possibly by program analysis. The Semfix and Angelix tools^{16,18} are a few examples of how the repair problem can be envisioned as one of inferring a repair constraint, and they have shown the scalability of such constraint-based techniques.

Conceptually speaking, automated program repair closes the gap between the huge manual effort spent today in writing correct programs, and the ultimate dream of generating programs automatically via learning approaches. Given the challenges of generating multiline program fixes in program repair, we can thus imagine the difficulty of generating explainable programs automatically.

Pragmatically speaking, automated program repair also makes us keenly aware of the challenges in managing changes in software engineering projects, and the need for automation in this arena. Today, manual debugging and maintenance often takes up 80% of the resources in a software project, prompting practitioners to long declare a legacy crisis.²⁵ In the future, program repair can provide tool support by repairing bugs from complex changes in software projects. This can help resolve a dilemma of developers when managing program changes: “Our dilemma is that we hate change and love it at the same time; what we really want is for things to remain the same but get better.”^d

Acknowledgments. The authors acknowledge many discussions with researchers at the Dagstuhl seminar 17022 on Automated Program Repair (Jan. 2017). Claire Le Goues acknowledges support of the U.S. National Science Foundation under grants no. CCF-1750116 and CCF-1738253. Michael Pradel acknowledges support of BMWF/Hessen within CRISP and support of the DFG within the ConCys and Perf4JS projects. Abhik Roychoudhury acknowledges support of National Research Foundation Singapore, National Cybersecurity R&D program (Award No. NRF2014NCR-NCR001-21). 

d Quote by Sydney J. Harris.

References

- Abadi, M., Budiu, M., Erlingsson, U. and Ligatti, J. Control-flow integrity. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, ACM, 340–353.
- Alur, R., Singh, R., Fisman, D. and Solar-Lezama, A. Search-based program synthesis. *Commun. ACM* 61 (2018), 84–93.
- Brown, D.B., Vaughn, M., Liblit, B. and Reys, T.W. The care and feeding of wild-caught mutants. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, (Paderborn, Germany, Sept. 4–8, 2017), 511–522.
- Fry, Z.P., Landau, B., and Weimer, W. A human study of patch maintainability. In *Proceedings of the Intern. Symp. on Software Testing and Analysis*, 2012, 177–187.
- Gupta, R., Pal, S., Kanade, A. and Shevade, S. DeepFix: Fixing common C language errors by deep learning. Assoc. for the Advancement of Artificial Intelligence, 2017.
- Harman, M. Automated patching techniques: The fix is in. *Commun. ACM* 53 (2010), 108–108.
- Johnson, B., Song, Y., Murphy-Hill, E. and Bowdidge, Z. Why don't software developers use static analysis tools to find bugs? In *Proceedings of the Intern. Conf. on Software Engineering*, 2013, 672–681.
- Jones, J.A., Harrold, M.J. and Stasko, J. Visualization of test information to assist fault localization. In *Proceedings of the ACM/IEEE Intern. Conf. on Software Engineering*, 9.
- Kim, D., Nam, J., Song, J. and Kim, S. Automatic patch generation learned from human-written patches. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, 2013.
- King, J.C. Symbolic execution and program testing. *Commun. ACM* 19 (1976).
- Logozzo, F. and Ball, T. Modular and verified automatic program repair. In *Proceedings of Object-Oriented Programming Systems Languages and Applications*, 2012.
- Long, F., Amidon, P. and Rinard, M. Automatic inference of code transforms for patch generation. In *Proceedings of the ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering*, 2017.
- Long, F. and Rinard, M. Staged program repair with condition synthesis. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015.
- Long, F. and Rinard, M. Automatic patch generation by learning correct code. In *Proceedings of the ACM Intern. Symp. on Principles of Programming Languages*, 2016.
- Marginean, A., Bader, J., Chandra, S., Harman, M., Jia, Y., Mao, K., Mols, A. and Scott, A. Sappix: Automated end-to-end repair at scale. In *Proceedings of the Intern. Conf. on Software Engineering*, Software Engineering in Practice track, 2019.
- Mechtaev, S., Yi, J. and Roychoudhury, A. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Proceedings of the ACM/IEEE Intern. Conf. on Software Engineering*, 2016.
- Monperrus, M. Automatic software repair: A bibliography. *ACM Computing Surveys* 51, 1 (2017).
- Nguyen, H.D.T., Qi, D., Roychoudhury, A. and Chandra, S. SemFix: Program repair via semantic analysis. In *Proceedings of the ACM/IEEE Intern. Conf. on Software Engineering*, 2013.
- Nistor, A., Chang, P.-C., Radoi, C. and Lu, S. Caramel: Detecting and fixing performance problems that have non-intrusive fixes. In *Proceedings of ICSE*, 2015.
- Perkins, J.H. et al. Automatically patching errors in deployed software. In *Proceedings of the Symp. on Operating Systems Principles*. ACM, 2009.
- Qi, Y., Mao, X., Lei, Y., Dai, Z. and Wang, C. The strength of random search on automated program repair. In *Proceedings of the Intern. Conf. on Software Engineering*, 2014.
- Ray, B., Hellendoorn, V., Godhane, S., Tu, Z., Bacchelli, A. and Devanbu, P. On the “naturalness” of buggy code. In *Proceedings of the 38th Intern. Conf. on Software Engineering* (Austin, TX, USA, May 14–22, 2016), 428–439.
- Sadowski, C., Aftandilian, E., Eagle, A., Miller-Cushon, L. and Jaspan, C. Lessons from building static analysis tools at google. *Commun. ACM* 61, 4 (Apr. 2018), 58–66.
- Samimi, H., Schäfer, M., Artzi, S., Millstein, T., Tip, F. and Hendren, L. Automated repair of HTML generation errors in PHP applications using string constraint solving. In *Proceedings of the 34th Intern. Conf. on Software Engineering*, 2012.
- Seacord, R., Plakosh, D. and Lewis, G. *Modernizing Legacy Systems: Software Technologies, Engineering Processes and Business Practices*. Addison Wesley, 2003.
- Shacham, O.M., Vechev, M.T. and Yahav, E. Chameleon: Adaptive selection of collections. In *Proceedings of Conf. on Programming Language Design and Implementation*, 2009, ACM, 408–418.
- Shriver, D., Elbaum, S. and Stolee, K.T. At the end of synthesis: narrowing program candidates. In *Proceedings of the Intern. Conf. on Software Engineering*, 2017.
- Singh, R., Gulwani, S. and Solar-Lezama, A. Automated feedback generation for introductory programming assignments. In *Proceedings of the Intern. Conf. on Programming Language Design and Implementation*, 2013.
- Smith, E.K., Barr, E., Le Goues, C. and Brun, Y. Is the cure worse than the disease? overfitting in automated program repair. In *Proceedings of the International Symposium on Foundations of Software Engineering (FSE)*, 2015.
- Su, Z. and Wassermann, G. The essence of command injection attacks in Web applications. In *Proceedings of Symp. on Principles of Programming Languages*, 2006, 372–382.
- Toffola, L.D., Pradel, M. and Gross, T.R. Performance problems you can fix: A dynamic analysis of memoization opportunities. In *Proceedings of Conf. on Object-Oriented Programming, Systems, Languages, and Applications*, 2015.
- Tufano, M., Watson, C., Bavota, G., Di Penta, M., White, M. and Poshvanyk, D. An empirical investigation into learning bug-fixing patches in the wild via neural machine translation. In *Proceedings of Intern. Conf. on Automated Software Engineering*, 2018.
- Urli, S., Yu, Z., Seinturier, L. and Monperrus, M. How to design a program repair bot? insights from the repairator project. In *Proceedings of Intern. Conf. on Software Engineering*, Track Software Engineering in Practice, 2018.
- Wei, Y., Pei, Y., Furia, C.A., Silva, L.S., Buchholz, S., Meyer, B. and Zeller, A. Automated fixing of programs with contracts. In *Proceedings of ACM Intern. Symp. on Software Testing and Analysis*, 2010.
- Weimer, W., Forrest, S., Kim, M., Le Goues, C. and Hurlley, P. Trusted software repair for system resiliency. In *Proceedings of 46th Annual IEEE/IFIP Intern. Conf. on Dependable Systems and Networks Workshops*, 2016.
- Weimer, W., Fry, Z. and Forrest, S. Leveraging program equivalence for adaptive program repair: Models and first results. In *Proceedings of ACM/IEEE Intern. Conf. on Automated Software Engineering*, 2013.
- Weimer, W., Nguyen, T.V., Le Goues, C. and Forrest, S. Automatically finding patches using genetic programming. In *Proceedings of ACM/IEEE Intern. Conf. on Software Engineering*, 2009.
- Xiong, Y., Liu, X., Zeng, M., Zhang, L. and Huang, G. Identifying patch correctness in test-based program repair. In *Proceedings of Intern. Conf. on Software Engineering*, 2018.
- Xuan, J., Martinez, M., Demarco, F., Clement, M., Marcote, S.L., Durieux, T., Le Berre, D. and Monperrus, M. Nopol: Automatic repair of conditional statement bugs in Java programs. *IEEE Trans. Software Engineering* 43, (2017).
- Yi, J., Ahmed, U.Z., Karkare, A., Tan, S.H. and Roychoudhury, A. A feasibility study of using automated program repair for introductory programming assignments. In *Proceedings of ACM SIGSOFT Intern. Symp. Foundations of Software Engineering*, 2017.

Claire Le Goues (clegoues@cs.cmu.edu) is an associate professor at Carnegie Mellon University, Pittsburgh, PA, USA.

Michael Pradel (michael@binaervarianz.de) is a professor at the University of Stuttgart, Germany.

Abhik Roychoudhury (abhik@comp.nus.edu.sg) is a professor at the National University of Singapore.

© 2019 ACM 0001-0782/19/12



Watch the authors discuss this work in the exclusive Communications video.
<https://cacm.acm.org/videos/automated-program-repair>

Novel approaches draw on the strength of game theoretic mechanism design.

BY MOSHE TENNENHOLTZ AND OREN KURLAND

Rethinking Search Engines and Recommendation Systems: A Game Theoretic Perspective

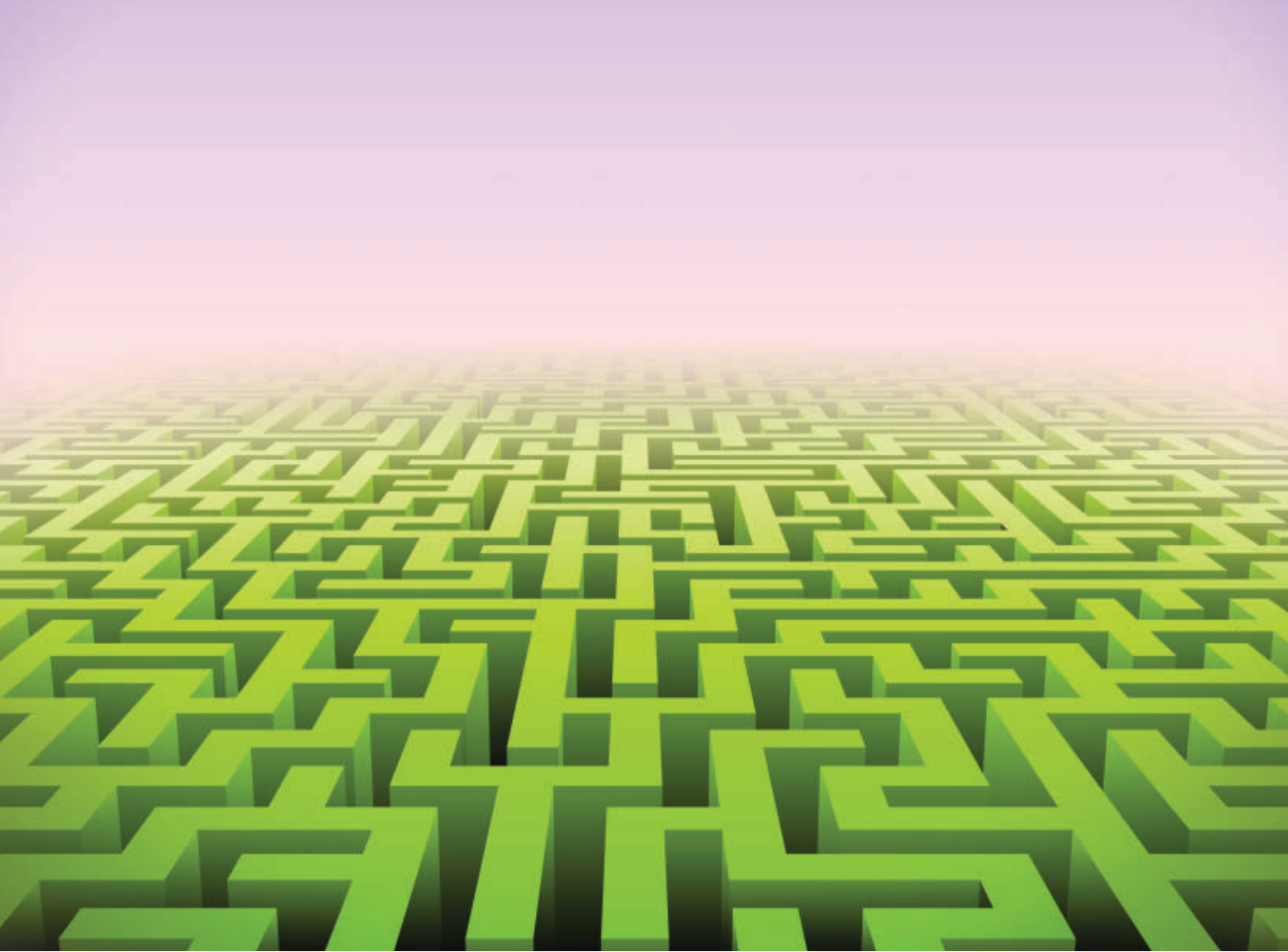
IN HER POPULAR book, *Weapons of Math Destruction*, data scientist Cathy O’Neil elegantly describes to the general population the danger of the data science revolution in decision making. She describes how the *US News* ranking of universities, which orders universities based on 15 measured properties, created new dynamics in university behavior, as they adapted to these measures, ultimately resulting in decreased social welfare. Unfortunately, the idea that data science-related algorithms, such as ranking, cause changes in behavior, and that this dynamic may lead to socially inferior outcomes, is dominant in our new online economy.

Ranking also plays a crucial role in search engines and recommendation systems—two prominent data science applications that we focus on in this article. Search engines (for example, Google and Bing) rank Web pages, images, and other items in response to a query. Recommendation systems endorse items by ranking them using information induced from some context—for example, the Web page a user is currently browsing, a specific application the user is running on her mobile phone, or the time of day.

In the industrial search and information retrieval community, the practice of adapting content to satisfy a search engine’s ranking function or that of a recommendation engine has been typically referred to as search engine optimization (SEO), which has become one of the most profitable professions in the data science era. Accordingly, there has been much work on devising algorithms that block spammers and guarantee that content presented to users is of high quality. Yet, virtually all retrieval and recommendation algorithms ignore post-ranking/recommendation effects (that is, manipulations applied to items) on the corpus. As it turns out, this reality results in underoptimized fundamental paradigms for devising search and recommendation algorithms as we discuss here. For example, a publisher of a Web page can try and mimic other Web pages to promote her page in rankings, thereby potentially causing content

» key insights

- An overall modeling and analysis of the search and recommendation systems ecosystems (publishers, users, mediators, and competitors) is essential for their efficient design.
- Strategic content dynamics may lead to failure of classical principles of search and recommendation systems in maximizing social welfare; these principles can/should be revisited.
- Game-theoretic mechanism design can be adapted to the design of economically efficient search and recommendation systems.



changes that are not for the better—for example, reducing content breadth in terms of topical coverage.

These observations call for action. We believe the design of search and recommendation methods central to the Web and to online media services, must be revolutionized to account for the strategic incentives of the various parties that affect (or are affected by) the system. The parties can be publishers/content providers, the systems themselves (for example, search engines) that might be in competition with each other, and/or users of the systems. We are in the process of establishing an entirely new repertoire of incentive-compatible search and recommendation algorithms. These are devised through pioneering the application of game theoretic mechanism design to these tasks.

Game theory is the branch of mathematics that models multiagent interactions. Mechanism design is the part of game theory that designs pro-

ocols/algorithms for environments consisting of self-motivated participants. Mechanism design has been central in bridging computer science and game theory,²⁹ including wide application to electronic commerce (for example, Cramton¹⁰), advertising (for example, Varian³⁸) and routing networks (Roughgarden³⁴).

In this article, we survey our recent work on creating theoretical foundations and developing empirically evaluated algorithms to build a fundamental bridge between game theory, and more specifically mechanism design, and search and recommendation. We argue it is only natural to model the competitive search and recommendation settings using game theory. For example, considering the system (search engine or recommendation system) as a mediator and those who produce items to be searched or recommended as players entails a suite of specific game settings. Items being highly ranked or

recommended amount to a high “pay off” for the owner of the item. Game-theoretic modeling allows to reason about the strategic behavior of players and the situations the setting can converge to (namely, equilibria). Furthermore, the task of devising search or recommendation algorithms becomes a mechanism design (or more specifically, mediator design) problem—a completely new view of these two important tasks.

The incentivized players we mainly focus on here are content providers/publishers who create and manipulate the retrieved/recommended items. In addition, we describe our work on addressing the competition between search engines—that is, where the engines are the incentivized players—and more generally, competition between prediction algorithms, focusing on regression; and addressing some of the implications of the strategic behavior of users of recommendation systems in social networks.

We discuss some of the fundamental, and far reaching, modeling and algorithmic implications of the competitive and incentivized settings in which search engines operate. Specifically, we examine post-ranking corpus effects and survey our work on a game-theoretic analysis of the strategic behavior of publishers (content providers) and on addressing ranking robustness. We also briefly describe the challenges of empirical evaluation and analysis in competitive search settings. Later, we focus on our theoretical work on the suboptimality in competitive settings of the most basic search (ranking) and recommendation principle and we survey our work on settings where incentivized parties are ranking algorithms and prediction algorithms (regression) that compete with each other.

Search Engines

The main goal of search engines is to rank documents in a corpus by their presumed relevance to the information need expressed by a query posted by a user. There has been a huge body of work on devising relevance estimates for documents. For example, the similarity between term-based vectors representing documents and queries serves as a relevance estimate in the well-known vector space model.³⁵ Modern relevance ranking functions are learned (trained) using a training set which contains queries and corresponding relevance judgments of documents.^{25,28} These approaches allow to integrate various types of relevance estimates.

The theoretical foundation of virtually all ad hoc retrieval paradigms, classical and modern, is the probability ranking principle (PRP).³³ Documents are ranked by the probability that they are relevant to the query wherein the probability is estimated using all the information available to the search system. The PRP is the direct basis of the probabilistic retrieval approach²⁰ and was shown to underlie the basic language modeling approach.²³ Feature-based learning-to-rank methods can also be viewed as obeying the PRP, even if relevance probability is not directly estimated. That is, these approaches are trained to minimize ranking loss with respect to ground truth ranking and they essentially integrate



We believe the design of search and recommendation methods must be revolutionized to account for the strategic incentives of the various parties that affect (or are affected by) the system.



various relevance estimates so as to improve overall relevance estimation that guides ranking. Neural-network-based retrieval methods estimate relevance (often probability) by learning document and query representations and/or integrating multiple relevance signals.²⁸ As feature-based approaches they can be viewed as obeying the PRP. The PRP was shown to be optimal under some assumptions; for example, the independence of document relevance from that of others.³³

Post-ranking dynamics. Careful examination of the PRP, and retrieval methods based on the PRP, reveals a major gap: post-ranking effects are not accounted for; specifically, changes in the corpus documents made by incentivized publishers that respond to the induced ranking so as to improve the ranking of their documents. We focus on publishers (content providers) as those affected by, and affecting, the dynamics. We do not consider the dynamics driven by users of the systems, for example, via clickthrough actions. Later in the article, we discuss the importance, challenges, and potential future directions of accounting simultaneously for publisher and user-driven dynamics.

To highlight the importance of accounting for post-ranking effects on corpus content consider the following example.² A publisher writes a document that contains both common information and information which is unique to the document; that is, this unique information cannot be found in any other document in the corpus. The publisher, for some reason, is more interested in her document being highly ranked for queries that touch on the non-unique information. However, the ranking function “penalizes” documents having a mixture of information (that is, not being focused on the information sought for in these queries). In terms of the PRP and current retrieval paradigms the approach of the ranking function is justifiable: one can assume that the user who posted the query would prefer reading documents that are focused on information related to the query rather than having to wade through potentially long documents and distill the sought information. Now, suppose that the publisher decides to remove

the unique information from the document so as to obtain higher ranking for queries pertaining to the common information. This means that valuable information in the corpus is lost, and search effectiveness for queries which target the unique information will consequently degrade. Overall, the potential satisfaction of users' information need with respect to the corpus will decrease.

Indeed, we showed, using a game theoretical analysis, that the PRP is suboptimal as it does not promote content breadth in the corpus in terms of topical coverage.² In other words, retrieval methods based on the PRP do not provide a strong enough incentive for publishers to produce diversified content. It also turns out that introducing randomness to ranking functions can help to promote content breadth in the corpus, and hence, increase the overall attainable utility for search engine users. We will discuss the sub-optimality of the PRP and the merits of using nondeterministic ranking functions later.

Given the significant impact of induced rankings on the content in the corpus, due to the actions employed by incentivized publishers who respond to these rankings, the natural challenge that rises is analyzing the strategic behavior of publishers. Previous work has characterized search engine optimization (SEO) techniques intended to promote documents in rankings.¹⁵ We also note that content dynamics on the Web was studied and analyzed (for example, Raifer et al.³² and Santos et al.³⁶). However, the post-ranking perspective has not been actually modeled or analyzed; that is, the specific types of responses of publishers to rankings, and more generally, their ranking-motivated strategies in terms of document manipulation, were not studied.

Strategic publishers. Since the early days of the Web, different types of SEO techniques have been identified.¹⁵ For example, publishers can “stuff” keywords in their Web pages so as to promote them in rankings induced for queries that include these keywords. The underlying (often correct) assumption is that increased similarity between the query and the Web page increases the retrieval score of the page,

and hence improves its potential rank position. On the other hand, compromising the quality of the page by filling it with potentially query relevant keywords can also reduce the retrieval score as ranking functions often utilize page-quality estimates.⁶

There is a fundamentally important question that goes beyond the actual general actions that publishers use as part of their SEO efforts: What is the strategic behavior of the publishers with respect to induced rankings? In other words, given that they do not know what the ranking function is,^a but they can observe past rankings induced for queries of interest, what would be an effective response strategy to rankings?^b We have recently addressed this question using a game theoretic analysis.³² Our main theoretical result was that a “worthwhile” strategy for publishers who want to promote their documents in rankings induced for a specific query is to make their documents become more similar to those highly ranked in the past for the query. By “worthwhile” we refer to the fact that this strategy results in a certain equilibrium in a game-theoretic modeling of the retrieval setting wherein publishers are players and the ranking function is a mediator, not exposed to the players, which induces rankings for queries.

The intuitive theoretical finding that mimicking the “winner” from previous rankings is worthwhile was supported by analyzing the strategic publishing behavior of students who served as publishers in a content-based ranking competition we organized. We provide details of this competition later.

Addressing unwarranted effects of the dynamics. As discussed, a major part of the dynamics of a corpus in competitive retrieval settings is driven by incentivized publishers. The dynamics can have undesirable effects, specifically, in terms of degrading retrieval effectiveness. For example, some Web pages could be spam that is intended

to be promoted in rankings and to attract clicks—that is, black-hat SEO.¹⁵ At the same time, corpus dynamics is driven to a major extent by white hat SEO efforts which need not necessarily hurt retrieval effectiveness; these are legitimate actions applied to Web pages so as to promote them in rankings. However, even white-hat SEO can lead to undesirable effects; for example, rapid changes to the relative ranking of documents due to indiscernible document manipulation. As a result, users might respond by consistently reformulating their queries or simply losing faith in the search engine.

Ranking robustness: A blessing or a curse? Following the arguments just posed, one should presumably opt for ranking robustness; that is, small indiscernible changes of documents should not result in major changes to induced rankings.¹⁴ Some support for this argument can be drawn using one of the most fundamental hypotheses in the field of information retrieval, namely, the cluster hypothesis¹⁸ as we recently described.¹⁴ Additional support can be drawn from arguments made in work on adversarial classification, specifically, in the vision domain. The main premise in this line of work was that small indiscernible (adversarial) changes of objects should not result in changes to classification decisions.^{11,13} Thus, while changes of rankings in our setting that reflect discernible changes of, or differences between, documents are naturally very important, changes of rankings that are “harder to explain” (a.k.a. explainable IR) are less warranted.

On the other hand, there are arguments for hurting ranking robustness, to some extent, for special purposes. For example, recent work showed that introducing randomization to induced rankings along time, which hurts robustness, can increase fairness with respect to publishers of Web pages.⁷ Specifically, given that users browsing the search results page mainly pay attention to the top ranked results,¹⁹ allowing Web pages to be positioned at the highest ranks even if they are not among the most relevant, can increase the attention given to other publishers and hence promote fairness. Another justification for hurting ranking robustness is promoting content

a There are efforts to reverse engineer ranking functions but these are often of quite limited success.

b A response in terms of manipulating document content can be, for example, at the “micro-level:” selecting the terms to add or remove, or at the “macro-level:” making a document more similar to another document.³²

breadth in the corpus, specifically, by introducing randomization to ranking functions, as we recently showed using a game theoretic analysis.²

Considering the findings described here, there is a trade-off between ranking robustness and content breadth and/or ranking fairness. Accounting for all these aspects simultaneously is an intriguing avenue for future work.

Robustness of ranking functions: Regularization and beyond. In recent work,¹⁴ we presented initial theoretical foundations for the analysis of the robustness of ranking functions to (adversarial) document manipulations intended for promoting the documents in rankings. While the goal was indeed to study adversarial effects on robustness, the formal treatment was more general and accounted for any type of a change.

An important theoretical result to which we provided empirical support was the connection between regularization—a suite of approaches intended to improve the effectiveness of a ranking function over queries not seen during the training phase—and ranking robustness. We showed that the “stronger” the regularization, the more robust the induced rankings are to document manipulations.

Controlling ranking robustness by tuning regularization is only a first step. Exploring other fundamental approaches to improving, or more generally, controlling ranking robustness is a completely open research avenue. While existing relevance ranking functions are trained to improve effectiveness or minimize reductions in effectiveness with respect to that of known ranking functions,³⁹ ranking robustness as a manifestation of post-ranking effects is a missing component in such approaches. The treatment of robustness of ranking functions is also important, as mentioned earlier, given the emergence of recent work on improving ranking fairness by hurting robustness.⁷ One would strive to simultaneously maximize fairness and minimize ranking instability.

Empirical analysis and evaluation. The Web is the canonical example of a competitive retrieval setting. Specifically, many publishers of Web pages have an incentive (for example, financial) to have their pages highly ranked for queries they care about. Isolating Web dy-

namics associated with incentive driven changes to documents is difficult since it occurs on a backdrop of constant Web page changes and dynamics that are not driven by ranking incentives.

To allow the study of corpus dynamics that results from the strategic behavior of publishers in response to rankings, as well as to devise new retrieval paradigms that account for this type of dynamics and its effects, controlled experiments are called for. For example, we recently reported a ranking competition held between students in a class.³² The students were instructed to write plain text documents with the aim of being highly ranked for given queries. That is, the competition was focused on content-based manipulation. The incentive of students to take part in the competition and produce documents that would presumably be ranked high was bonus points for the course grade if they were to win many rounds of the competition; that is, if the documents they created were highly ranked. At the end of each round of the competition, the students were shown the rankings induced for queries over the documents they have created/changed. That is, the only information available for “publishers” in this competition, as is the case over the Web, is the query, the induced ranking, and the ranked documents.

The competition was not large scale and focused on a specific (yet important) aspect of manipulation—namely, content manipulation. Nevertheless, as mentioned earlier, it provided empirical support to a game theoretical result we presented about the strategic behavior of publishers in response to induced rankings:³² publishers make their documents similar to those that were highly ranked for the same queries in previous rounds of the competition.

Organizing ranking competitions at large scale is extremely difficult, specifically, if various aspects are to be studied simultaneously—for example, the manipulation of both content and hyperlink structure. Furthermore, to develop novel ranking functions, one must run A/B testing between old and new functions in real time—that is, to have publishers participating in the game respond, by manipulating their documents, to the ranker which is ap-

plied. Thus, analysis of ranking competitions between incentivized publishers, as well as evaluation of novel ranking methods that address the resulting corpus dynamics, is a challenge at its own right.

Non-Optimality of Classical Search and Recommendation Methods

As noted, the probability ranking principle (PRP) is the theoretical foundation of most retrieval approaches that rank documents in response to a query. It is, in essence, also the theoretical foundation of recommendation methods that rank and recommend items. Accordingly, aside for potential exploration intended to collect data for training, search and recommendation methods based on the PRP are deterministic. As it turns out, when the publishers of items to be searched or recommended are strategic, this classical deterministic approach has to be replaced. Here, we consider the PRP and approaches devised based on the PRP for search (retrieval) and recommendation, respectively, in settings with strategic publishers.

Beyond PRP: Randomness increases social welfare in search. One of the important goals of an analysis of a competitive retrieval setting with corpus dynamics driven by incentivized (strategic) selfish players—publishers, in our case—is to derive insightful statements about the steady states that the setting can converge to, if at all. It is important to note that here, dynamics refers to that of documents in the corpus upon which search is performed. Dynamics of clickthrough patterns^c and that of users’ queries⁴⁰ is outside the scope of this article. To address this goal, it is only natural to take a game theoretic approach that allows reasoning about the different equilibria the search setting can converge to. Equilibrium is a steady state wherein players (publishers) have no incentive to deviate from the current actions (strategies) they employ.^d The fundamental question then becomes how to model a retrieval setting as a game.

c Clickthrough refers to clicks of users on the results presented by the search engine.

d These strategies can be probability distributions (a.k.a. mixed strategies) over pure strategies.

We have recently modeled competitive retrieval settings as games.^{2,3} Publishers are players whose strategies are the types of documents they can write: either single-topic or multi-topic documents. The search engine's ranking function is the mediator that induces a ranking. A publisher is rewarded if her document is the highest ranked for a query. To simplify the analysis, we assumed each query is about a single topic and the distribution over queries is known to the publishers. We then examined two settings: all publishers have the same equal writing quality over all topics, and publishers have differential writing quality for different topics.

The type of documents publishers can write and the assumption regarding their writing quality entails a specific game. We assumed the ranking function has full knowledge of whether a document is relevant to a query (specifically, by topic matching) and analyzed the equilibria of the game, that is, publishers' behavior in which modification of behavior will not be any unilaterally beneficial. We then analyzed the social welfare attained in the game, specifically, in various equilibria. Social welfare was defined as the sum of utilities attained by publishers. We assumed users' utilities and publishers' utilities were aligned and those were determined by whether the highest ranked document was relevant. A simple utility function was used: 1 if the highest-ranked document is relevant and 0 otherwise.^{2,3} This utility function corresponds to a setting where a user examines only the top-ranked document and her utility solely depends on the relevance of this document. The alignment between publishers' and users' utilities facilitates the formal treatment and corresponds to the assumption that if a user is satisfied by seeing a relevant document then the publisher of the document is rewarded to the same extent. In practice, however, publishers' and users' utilities are not necessarily aligned, and more evolved models are thereby called for.^{2,3}

The games were further analyzed by computing the price of anarchy;^{21,34} the ratio between the maximal social welfare that can be attained in a game and the minimal social welfare attained in

A Simple Example of the Non-Optimality of the PRP

A simple way to demonstrate the suboptimality of the probability ranking principle (PRP) in a game-theoretic setting is as follows: Assume two authors (players 1 and 2) who can write on two topics—that is, each player chooses between two strategies: writing on topic *A* or writing on topic *B*. There is big user demand, $U(A)$, to topic *A*, and a slightly lower user demand $U(B) = U(A) - \epsilon$, to topic *B*. (We assume a query is about a single topic and the demand represents the distribution over queries.) Player 1 can write with optimal quality (1) on both topics, while player 2 can write with close to optimal quality $(1 - \epsilon)$ on topic *A* but with really poor quality (ϵ) on topic *B*. The authors' gain (utility) is the quality of their writing times the demand they get on the topic for which they are ranked the highest. The total users' satisfaction (social welfare) however is the overall total quality over both topics. According to the PRP, the higher-quality document is ranked first for a query for each topic. In this game, the only equilibrium is for player 1 to write on topic *A*, and for player 2 to write on topic *B*; in this case the total users' satisfaction is not much more than half of the total social welfare (satisfaction) the users could get (that is, the price of anarchy is slightly less than 2). The PRP simply gives no chance to player 2, who could be writing so well on topic *A*, although this is the only topic he writes well about.

any equilibrium of a game. In other words, price of anarchy is the price the echo system “pays” for the selfish behavior of players. The accompanying sidebar describes an example of a game and its analysis.

We found that if publishers write single-topic documents and they have equal writing qualities for all topics then the PRP is indeed an optimal criterion for ranking.^{2,3} But, if publishers write multitopic documents or have differential writing qualities the PRP becomes suboptimal in terms of price of anarchy. The sidebar provides a simple example that demonstrates the suboptimality of the PRP. Indeed, the PRP motivates publishers to write single-topic documents, or documents only on the topics they are most qualified to write, as their resultant ranking would be higher. Thus, the PRP essentially does not promote content breadth in the corpus—in terms of topical coverage—in these natural settings. Previously, we described a canonical example of this situation: the publisher of a multitopic document, with information unique to the document, is driven by the PRP to remove the unique information so as to better focus the document for higher ranking for queries of interest.

It turns out that introducing randomization to the ranking functions can improve the price of anarchy with respect to that attained by using the PRP as a ranking criterion, as random-

ization can help to promote content diversity.^{2,3} A case in point, promoting in rankings a multitopic document with respect to a single topic document, in case the retrieval score of the latter is a bit higher than that of the former, does not significantly harm per-query retrieval effectiveness but does result in improved price of anarchy (and social welfare) due to increased content breadth in the corpus.

These findings are especially important because they imply the most fundamental principle used by existing retrieval methods and search engines for ranking (the PRP) is suboptimal in competitive retrieval settings. Furthermore, these findings motivate a new view of how ranking functions should be learned: not (only) for minimizing a “local” loss function but rather maximizing the resultant social welfare attained in equilibria. This is a brand new challenge for the information retrieval community, and other communities interested in ranking, as the task involves estimating post-ranking effects. Our findings about the merits of applying nondeterministic ranking functions is a first small step in this direction.

Characterizing a desired fair recommendation system. Heretofore, we have mainly focused on search (retrieval) systems. Recommendation systems (RSs hereinafter) have also rapidly developed over the past decade. By predicting a user preference for an item,

RSs have been successfully applied in a variety of applications. However, in order to address significant applications, where information is integrated from many sources, such systems have to generate recommendations that satisfy the needs of both the end users and other parties, and in particular content providers (publishers). Many of the content providers provide sponsored or partially sponsored content that may be relevant to a user, and an RS that will not be fair with content providers, will not survive.

Hence, an RS should be able to deal in a fair way with strategic content providers, each of which aims at maximizing its exposure. In addition, if an RS is not designed properly, then content providers may reach a fluctuating system where content providers rapidly change the type of content they provide only to defeat their competitors. Therefore, the RS need not only be fair but to also yield stability, that is, convergence to equilibrium of content providers' strategies. These RS requirements were only formulated very recently in a rigorous manner.⁵ It has been shown that classical RSs, which operate in the PRP style (that is, always show the most relevant content) fail to satisfy the requirements noted here (that is, will be unfair or lead to fluctuations). Indeed, the latter work also shows a particular RS, selecting among content providers in a well-defined probabilistic manner, that does satisfy the related requirements, and in a sense is the only recommendation system that satisfies them. Technically, the results use a classical concept in cooperative game theory, the Shapley value,³⁷ which measures the contribution of parties to a cooperative act, and adapt it to serve as an efficient mechanism for probabilistic selection among content providers.

These findings complement the observation about the failure of the most fundamental principle used by existing retrieval methods and search engines for ranking (the PRP), by showing that the corresponding method also fails in the context of recommendation systems. It is suboptimal when considering strategic content providers. Indeed, well-defined

and well-grounded probabilistic substitutes are offered.

Dueling

So far we have dealt with the design of systems, a.k.a. mechanisms or mediators, while taking into account that participants in these systems (specifically, publishers/content generators) are self-motivated. Indeed, the role of a search engine, as well as the role of a RS, which integrate information from different content providers, is to serve as a mediator among selfish participants. However, such mechanisms may also face a competition with other mechanisms as we will discuss. Thus, we now describe models that address such competitions (specifically, between search engines/ranking functions and between prediction algorithms). In these settings, the systems themselves can be thought of as the players, in contrast to the settings discussed thus far where publishers were the players.

Consider two competing search engines. A user thinks of a desired Web page and submits a query intended to target such a page to both search engines. The engine that ranks the desired page higher is chosen by the user as the "winner." Given a probability distribution over the users as for their desired pages given that query, the optimal algorithm ranks the pages, $\omega_1, \omega_2, \dots, \omega_n$, in a decreasing order of the corresponding probabilities; this is essentially the probability ranking principle (PRP). However, in a competitive setting the ranking $\omega_2, \omega_3, \dots, \omega_n$, ω_1 wins (assuming the proportion of users interested in ω_1 is less than 0.5) on every item that may be searched except for ω_1 . This is a highly simplified example of the fact that different data science mechanisms are in competition with one another, and therefore the optimal way to design them should be reconsidered.

Such setting was formalized first in Immorlica et al.,¹⁶ under the title *dueling algorithms*. The most basic building block in such settings is to consider a state-of-the-art algorithm (for example, a ranker in the information retrieval setting, a regressor in a prediction setting) and see if it can be defeated when put in a competitive setting; such as, whether one can pro-

vide an algorithm that most users will find more compelling than the classical one, when they have to choose. We now discuss two achievements that have been obtained in this regard.

Best-response regression. An interesting extension of this general discussion is in the context of machine learning, and in particular regression tasks. In a regression task, a predictor is given a set of instances (points) along with a real value for each point. Subsequently, she must identify the value of a new instance as accurately as possible. Ben-Porat and Tennenholtz⁴ considered a regression task tackled by two players, where the payoff of each player is the proportion of the points she predicts more accurately than the other player. On the technical side, in order to do so, the authors revise the probably approximately correct (PAC) learning framework to deal with the case of a duel between two predictors. Then they devise an algorithm that finds a linear regression predictor that is a best response to any (not necessarily linear) regression algorithm. This algorithm has linearithmic sample complexity, and polynomial time complexity when the dimension of the instances domain is fixed. The approach has been also tested in a high-dimensional setting, and it has been shown to significantly defeat classical regression algorithms in the prediction duel.

Responding to a strong ranker. Returning to the search and ranking motivation, here is a fundamental interesting question. Say a search engine has a relatively weak (somewhat ineffective) relevance ranking function; specifically, with respect to that of another (strong) search engine. This could be, for example, due to a relatively small user base; indeed, user engagement signals are important features in relevance-ranking functions.

Thus, an interesting duel between the two search engines emerges. The duel entails an interesting bimodal utility function for the weak search engine: the goal is to produce in response to a query a document result list whose effectiveness is not significantly lower than that of the strong search engine; and, the result list should also be quite different than that produced by the strong engine. In Izsak et al.,¹⁷ we

presented a per-query algorithmic approach that leverages fundamental retrieval principles such as pseudo-feedback-based relevance modeling.²⁴ That is, the weak ranker observes the documents most highly ranked by the strong ranker, uses them to induce a model of relevance for the query, and uses this model for ranking. We demonstrated the merits of our approach in improving the search effectiveness of the weak ranker using TREC data^e—a suite of datasets used for evaluation of retrieval methods.

Strategic Users

The previous sections emphasized the need to account for the incentives of strategic parties when devising major data science applications such as search (ranking), recommendation, and prediction (specifically, regression). The strategic parties were associated with stakeholders such as content providers or prediction experts. The users of the systems were not strategic; they were the ones posting a query, asking for recommendation, or interested in prediction. In a sense, the users were the products the strategic parties were aiming to attract or to serve better than their competitors. However, users may have their own incentives that might cause them to potentially not follow recommendations by the system or to provide dishonest inputs to the system. Next, we briefly describe one of our recent works that addresses such aspects.

Incentive-compatible explore and exploit. Recommendation systems based on the interleaving of exploration and exploitation paradigm have a two-way relationship with their customers—on the one hand they provide recommendations while on the other hand they use customers as their source of information. This reality leads to an important challenge: a user might not accept a recommendation if he/she believes the recommendation is done to benefit exploration rather than be the optimal one given current information. The challenge is to devise a system that will behave close to optimal, but will be also incentive compatible, that is,



A recommender system should be able to deal in a fair way with strategic content providers, each of which aims at maximizing its exposure.



rational users who care only for their expected utility will accept the system's recommendations.

We have recently addressed this challenge in designing recommendation systems, specifically for social networks, where users can (partly) observe each other.^{1,f} In particular, we investigated the conditions on the social network that allow for asymptotically optimal outcomes. Our results show that for reasonable networks, where each user can observe many of the other users, but where still most users cannot see most of the other users, an incentive compatible and approximately optimal recommendation system does exist.

The literature on incentivizing exploration relevant to our study can be viewed as an extension of the celebrated multi-armed bandit problems⁸ to deal with settings where exploration can only be done by self motivated myopic agents and a central planner must incentivize exploration. The tension between the objective of the mediator (the recommendation engine) and the individual agents in a multi-armed bandit context was first introduced in Kremer et al.,²² who study this in a very simple setting. They identify an incentive compatible scheme with which a central mediator with commitment power can asymptotically steer the users toward taking the optimal action. This exciting news has been extended in Mansour et al.^{26,27} to several more elaborate bandit settings and to additional optimization criteria such as regret minimization. Whereas both of these papers account for agents' incentives and in particular the misalignment of incentives of the agents and the mediator they ignore other societal aspects. In particular, these papers make an implicit assumption that agents cannot see nor communicate with any other agent. Needless to say, the assumption that agents have no knowledge of each other, and cannot see the actions chosen by their neighbors, is unrealistic. Che and Horner⁹ also study mechanisms for social learning through a mediator. Similar to Kremer et al.,²² they assume agents are short lived and do not observe each other and the mechanism

^f The pioneering work on this challenge assumed no user communication.^{9,22}

^e <https://trec.nist.gov/>

must incentivize them to explore. In contrast, the model in Che and Horner⁹ is one of continuous time.

Unfortunately, the results obtained in Kremer,²² and Mansour^{26,27} are not robust to changes in such network assumptions and their implicit assumption of no visibility turns out to be critical. In fact, even with very little observability, for example when each agent just sees the action chosen by his immediate predecessor, the schemes proposed in both works cease to be incentive compatible and lead to market failure. The challenge of approximately optimal incentive compatible schemes subject to partial observability among the agents was addressed in our work.¹ In that paper it is assumed agents can view actions chosen by some of their predecessors. We characterize structural properties that allow for the design of optimal schemes and provide explicit constructions. In contrast with the recent literature relevant to search and recommendation where payments are avoided, other works study payments to agents as a means to incentivize exploration (for example, Frazier et al.¹²).

Conclusion and Looking Forward

Today, many search engines and recommendation systems operate in dynamic settings wherein dynamics is driven by interested parties. The parties can be those that generate items to be retrieved or recommended, the users themselves or competing systems.

It is thus somewhat surprising there has been little to no work on modeling the incentives of parties involved, and more generally, on devising algorithms that account for the incentive driven dynamics of the ecosystem they operate in. We discussed our results regarding the consequences of ignoring the dynamics in several cases: the attained solutions are, by design, suboptimal. In particular, we showed that the most basic principle underlying most query-based ranking (retrieval) algorithms and recommendation methods is suboptimal in competitive settings.

Accordingly, we argued for a need to revolutionize the way search and recommendation algorithms are devised. We suggested game theory, and more spe-



Our ongoing and ultimate operational goal is to devise new and improved search and recommendation algorithms via mechanism design.



cifically mechanism design, as a framework for devising the algorithms. The basic idea is to treat interested/incentivized parties as players and the data system/algorithm as a mediator. The challenge is that of mechanism (mediator) design: devising an algorithm that accounts for the environment which consists of self-motivated parties.

An interesting question is whether the game theoretic models constitute effective approximations for real user behavior. In our recent work we demonstrated such an alignment:³² the publishing strategies of publishers were aligned with the game-theoretic results; namely, mimicking documents which were highly ranked in the past. Studying the robustness of game-theoretic solutions in the face of irrational behavior of users is a direction we intend to explore in future work.

Another important aspect is the system and algorithmic complexity overhead when accounting for incentives and applying game theoretic principles. We argue that the overhead need not be significant. For example, the nondeterministic ranking functions we proposed^{2,3} improve social welfare with respect to the PRP and practically post no algorithmic or system-implementation overhead. Similarly, our approaches for improving ranking robustness¹⁴ and for dueling of search engines¹⁷ incur no significant overhead.

Our ongoing and ultimate operational goal is to devise new and improved search and recommendation algorithms via mechanism design. We surveyed a few such examples, but there is much more work to be done in that respect. We envision a modular development of game theoretic-based search and recommendation solutions, as was and is the case for current solutions that ignore incentives. First, fundamental relevance estimates for search and recommendation will be improved by mechanism design and these can be integrated—for example, via learning-to-rank—with other estimates for which incentive issues are not accounted for. At the macro level, one can devise loss functions for search and recommendation functions that account for incentives. These can be devised in some cases, as we recently showed for ranking robustness,¹⁴ regardless

of the features used. Overall, such modular development allows, in many cases, to separate the development of none-game-theoretic aspects from those which rely on game theory.

The work we surveyed (for example, on nondeterministic ranking functions^{2,3} provides some initial guidelines about how one would devise solutions that account for incentives. We believe the suite of guidelines will broaden with more work in these directions, as is the case, for example, for neuro IR—current state-of-the-art neuro IR algorithms are significantly different from those proposed a few years ago given the experience which has been gained.

The spectrum of challenges that remain to be addressed is huge. For example, accounting for the interactions between different types of incentivized players involved—for example, publishers and users of a search engine. Indeed, users of the search engine are not only incentivized consumers of retrieved items, but also interested parties that affect the search engine's behavior (specifically, its ranking function) via their interactions. That is, in commercial search engines, users' clicks on retrieved results are treated as implicit relevance feedback.¹⁹ Query reformulations are an additional implicit relevance feedback signal. Thus, modeling user interactions with the search system is an important part of the analysis of any search engine and the design of retrieval algorithms. In our setting, the emerging challenge is modeling simultaneously, using a game theoretic approach, the effects of the strategic behavior of interested users with that of interested publishers. For example, to contrast ranking mechanisms in terms of utility and social welfare, we can potentially use Bayesian games, rather than perfect information games used in our initial work,^{2,3} wherein clicks and their dynamics are employed to devise relevance estimates.

There are additional “micro-level” challenges; examples include devising content-based and other (for example, hyperlink-based) relevance estimates that account for incentivized publishers; moving from list-based effectiveness evaluation to amortized and equilibrium-based evaluation^{3,7} given the dynam-

ics of the retrieval setting; and, adapting specialized retrieval approaches (for example, those based on fusion or federation) to incentives-based settings.

Our study is part of a more general attack on applying mechanism design for data science (see <http://mdds.net.technion.ac.il>). Indeed, while algorithmic game theory had remarkable success in addressing incentives in variety of settings, such as a variety of fundamental resource allocation settings (for example, auctions), and congestion control,²⁹ among others, both the repertoire of games and the techniques to be employed need to be expanded in order to deal with data science algorithms. While we referred mainly to search and recommendation systems in this article, the site mentioned here shows applications to prediction, regression, segmentation, diffusion, and others. Two of the major techniques used are around approximate mechanism design without money³⁰ and dueling algorithms,¹⁶ both of which are beyond the scope of this article. **C**

References

1. Bahar, G., Smorodinsky, R., and Tennenholtz, M. Economic recommendation systems: One page abstract. In *Proceedings of EC*, 2016, 757–757.
2. Ben-Basat, R., Tennenholtz, M., and Kurland, O. The probability ranking principle is not optimal in adversarial retrieval settings. In *Proceedings of ICTIR*, 2015, 51–60.
3. Ben-Basat, R., Tennenholtz, M., and Kurland, O. A game theoretic analysis of the adversarial retrieval setting. *J. Artif. Intell. Res.* 60 (2017), 1127–1164.
4. Ben-Porat, O. and Tennenholtz, M. Best response regression. In *Proceedings of NIPS*, 2017, 1498–1507.
5. Ben-Porat, O. and Tennenholtz, M. A Game-theoretic approach to recommendation systems with strategic content providers. In *Proceedings of NeurIPS*, 2018, 1118–1128.
6. Bendersky, M., Croft, W.B., and Diao, Y. Quality-biased ranking of Web documents. In *Proceedings of WSDM*, 2011, 95–104.
7. Biega, A.J., Gummadi, K.P., and Weikum, G. Equity of attention: Amortizing individual fairness in rankings. In *Proceedings of SIGIR*, 2018, 405–414.
8. Bubeck, S. and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning* 5, 1 (2012), 1–122; <https://doi.org/10.1561/22000000024>
9. Che, Y.K. and Horner, J.O. Optimal Design for Social Learning. Cowles Foundation Discussion Paper No. 2000, 2015.
10. Cramton, P., Shoham, Y., and Steinberg, R. An overview of combinatorial auctions. *SIGecom Exchanges* 7, 1 (2007), 3–14.
11. Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. Adversarial classification. In *Proceedings of KDD*, 2004, 99–108.
12. Frazier, P.I., Kempe, D., Kleinberg, J.M., and Kleinberg, R. Incentivizing exploration. In *Proceedings of EC*, 2014, 5–22.
13. Goodfellow, I.J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*, 2015.
14. Goren, G., Kurland, O., Tennenholtz, M., and Raiber, F. Ranking robustness under adversarial document manipulations. In *Proceedings of SIGIR*, 2018, 395–404.
15. Gyöngyi, Z. and Garcia-Molina, H. Web spam taxonomy. In *Proceedings of AIRWeb*, 2005, 39–47.
16. Immorlica, N., Kalai, A.T., Lucier, B., Moitra, A., Postlewaite, A., and Tennenholtz, M. Dueling algorithms. In *Proceedings of STOC*, 2011, 215–224.
17. Izsak, P., Raiber, F., Kurland, O., and Tennenholtz, M. The search duel: A response to a strong ranker. In *Proceedings of SIGIR*, 2014, 919–922.
18. Jardine, N. and van Rijsbergen, C.J. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval* 7, 5 (1971), 217–240.
19. Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of SIGIR*, 2005, 154–161.
20. Jones, K.S., Walker, S., and Robertson, S.E. A probabilistic model of information retrieval: development and comparative experiments—Part 1. *Information Processing and Management* 36, 6 (2000), 779–808.
21. Koutsoupias, E. and Papadimitriou, C. Worst-case equilibria. In *Proceedings of STACS*, 1999.
22. Kremer, I., Mansour, Y., and Perry, M. Implementing the wisdom of the crowd. *J. Political Economy* 122 (2014), 988–1012.
23. Lafferty, J. and Zhai, C. Probabilistic relevance models based on document and query generation. *Language Modeling and Information Retrieval*. Kluwer Academic Publishers, 2003, 1–10.
24. Lavrenko, V. and Croft, W.B. Relevance-based language models. In *Proceedings of SIGIR*, 2001, 120–127.
25. Liu, T. *Learning to Rank for Information Retrieval*. Springer, 2011.
26. Mansour, Y., Slivkins, A., and Syrgkanis, V. Bayesian incentive-compatible bandit exploration. In *Proceedings of EC*, 2015.
27. Mansour, Y., Slivkins, A., Syrgkanis, V., and Wu, Z.S. Bayesian Exploration: Incentivizing Exploration in Bayesian Games. *CoRR* (2016); <http://arxiv.org/abs/1602.07570>
28. Mitra, B. and Craswell, N. Neural Models for Information Retrieval. *CoRR* (2017); <http://arxiv.org/abs/1705.01509>
29. Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V.V. *Algorithmic Game Theory*. Cambridge University Press, 2007.
30. Procaccia, A. and Tennenholtz, M. Approximate mechanism design without Money. In *Proceedings of EC*, 2009.
31. Radinsky, K. and Bennett, P.N. Predicting content change on the Web. In *Proceedings of WSDM*, 2013, 415–424.
32. Raifer, N., Raiber, F., Tennenholtz, M., and Kurland, O. Information retrieval meets game theory: The ranking competition between documents' authors. In *Proceedings of SIGIR*, 2017, 465–474.
33. Robertson, S.E. The Probability Ranking Principle in IR. *J. Documentation* (1977), 294–304. Reprinted in *Readings in Information Retrieval*, K. Sparck Jones and P. Willett (eds), 1997, 281–286.
34. Roughgarden, T. and Tardos, E. How bad is selfish routing? *JACM* 49, 2 (April 2002), 236–259.
35. Salton, J., Wong, A., and Yang, C.S. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (Nov. 1975), 613–620.
36. Santos, A.S.R., Pasini, B., and Freire, J. A first study on temporal dynamics of topics on the Web. In *Proceedings of WWW*, 2016, 849–854.
37. Shapley, L.S. A value for n-person games. *Contributions to the Theory of Games II*, Harold W. Kuhn and Albert W. Tucker, (eds.), Princeton University Press, Princeton, NJ, 1953, 307–317.
38. Varian, H.R. Online ad auctions. *American Economic Review* 99, 2 (2009).
39. Wang, L., Bennett, P.N., and Collins-Thompson, K. Robust ranking models via risk-sensitive optimization. In *Proceedings of SIGIR*, 2012, 761–770.
40. Yang, G.H., Sloan, M., and Wang, J. *Dynamic Information Retrieval Modeling*. Morgan & Claypool Publishers, 2016.

Moshe Tennenholtz (moshet@ie.technion.ac.il) is a professor in the Faculty of Industrial Engineering at Management at Technion, Haifa, Israel, where he holds the Sondheimer Technion Academic Chair.

Oren Kurland (kurland@ie.technion.ac.il) is a professor in the Faculty of Industrial Engineering at Management at Technion, Haifa, Israel.

ACM Transactions on Quantum Computing (TQC)

Open for
Submissions

Publishes high-impact, original research papers and select surveys on topics in quantum computing and quantum information science



Recent advances in quantum computing have moved this new field of study closer toward realization and provided new opportunities to apply the principles of computer science. A worldwide effort is leveraging prior art as well as new insights to address the critical science and engineering challenges that face the design, development, and demonstration of quantum computing. Alongside studies in physics and engineering, the field of quantum computer science now provides a focal point for discussing the theory and practice of quantum computing.

ACM Transactions on Quantum Computing (TQC) publishes high-impact, original research papers and select surveys on topics in quantum computing and quantum information science. The journal targets the quantum computer science community with a focus on the theory and practice of quantum computing including but not limited to: quantum algorithms and complexity, models of quantum computing, quantum computing architecture, principles and methods of fault-tolerant quantum computation, design automation for quantum computing, issues surrounding compilers for quantum hardware and NISQ implementation, quantum programming languages and systems, distributed quantum computing, quantum networking, quantum security and privacy, and applications (e.g. in machine learning and AI) of quantum computing.

For more
information
and to submit
your work,
please visit:

tqc.acm.org



Association for
Computing Machinery

research highlights

P. 78

Technical Perspective Bootstrapping a Future of Open Source, Specialized Hardware

By Michael B. Taylor

P. 79

OpenPiton: An Open Source Hardware Platform For Your Research

By Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen,
Yanqi Zhou, Alexey Lavrov, Mohammad Shahradd, Adi Fuchs,
Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff

Technical Perspective

Bootstrapping a Future of Open Source, Specialized Hardware

By Michael B. Taylor

COMPUTER ARCHITECTURE IS currently undergoing a radical and exciting transition as the end of Moore's Law nears, and the burden of increasing humanity's ability to compute falls to the creativity of computer architects and their ability to fuse together the application and the silicon. A case in point is the recent explosion of deep neural networks, which occurred as a result of a drop in the cost of compute because of successful parallelization with GPGPUs (general-purpose graphics processing units) and the ability of cloud companies to gather massive amounts of data to feed the algorithms. As improvements in general-purpose architecture slow to a standstill, we must specialize the architecture for the application in order to overcome fundamental energy efficiency limits that prevent humanity's progress. This drive to specialize will bring another wave of chips with neural-network specific accelerators currently in development worldwide, but also a host of other kinds of accelerators, each specialized for a particular planet-scale purpose.

Organizations like Google, Microsoft, and Amazon are increasingly finding reasons to bypass the confines imposed by traditional silicon companies by rolling their own silicon that is tailored to their own datacenter needs. In this new world, a multicore processor acts more of a caretaker of the accelerator rather than the main act.

However, specialization brings challenges, primarily the high NRE (non-recurring engineering) costs, and the long time-to-market of developing customized chips. Ultimately this NRE will limit the diversity of specialized chips that are economically feasible. For this reason, a new style of computer architecture research has emerged, which attacks the challenge of driving down the cost and time to market of developing these specialized hardware designs. A growing movement within academia

is to train Ph.D. students with the skills necessary to succeed in this brave new world, learning not only how to perform research but also to design and build chips. Both feeding into and out of this approach is the growth of an active open source movement that ultimately will provide many of the components that will be mixed and matched to create low-NRE designs.

The OpenPiton research, led by Princeton professor David Wentzlaff, is one of the watershed moments in this fundamental shift toward the construction of an open source ecosystem in the computer architecture. OpenPiton is an open source distributed cache-coherent manycore processor implementation for cloud servers. Unlike most multicore implementations, OpenPiton implements a new scalable, directory-based cache-coherence protocol (known as P-Mesh) with three levels of cache, including a distributed, shared last-level L2 cache that scales in size with the number of tiles. Cache coherence is maintained using three physical Networks on Chip (NoCs), which can connect general-purpose cores, accelerators, and other peripherals, and can be extended across multiple chips to build systems with up to 500 million cores.

In contrast to existing Intel Xeon processors, manycores are designed to have low implementation complexity, in order to minimize NRE. Manycore is clearly the future of low-NRE general-purpose server architecture and provides scalable general-purpose performance that can be coupled with emerging classes of accelerators. The OpenPiton work advances the bar not only by releasing open source for use by others, but also serving as a framework for micro-architectural exploration in Infrastructure-as-a-Service (IaaS) clouds. Much of this micro-architectural work focuses on how resources, whether inside a core, in the cache, or in the on-chip or off-

chip interconnect, are shared between different jobs running on the system. Other OpenPiton-related work has also explored issues in security and side-channel attacks.

While most computer architects use in-house simulators or from-scratch implementations to do their research, which results in questionable claims of validity and reproducibility, the Princeton team took an extremely clever approach: they leveraged an existing open source processor design, the OpenSPARC T1, to extend it into an entirely new scalable design. Then, the team integrated their research projects into this chip design, and taped out many of their research projects, so that all of these have a real-world physical realization in an advanced 25-core manycore processor in 32-nm technology. Then, they realized this effort as the open source OpenPiton scalable processor, which is the only real-world open source platform for both prototyping and experimenting with Linux-capable manycore.

I believe this work will unlock the next 20 years of progress in Linux-capable manycore research in academia, which has largely fizzled because of the lack of realistic, silicon-validated models to work with. At the same time, OpenPiton's scalable cache coherence implementation is licensed under the BSD license, which allows it to be freely mixed and matched. Indeed, work is already underway to retrofit open source RISC-V implementations like BlackParrot and Ariane. I expect OpenPiton's influence will grow across the community and enable larger and larger research projects that can truly deliver specialization across the stack. 

Michael B. Taylor is a professor in the Paul Allen School of Computer Science and Engineering and the Department of Electrical Engineering at the University of Washington, Seattle, WA, USA.

Copyright held by author/owner.

OpenPiton: An Open Source Hardware Platform For Your Research

By Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff

Abstract

Industry is building larger, more complex, manycore processors on the back of strong institutional knowledge, but academic projects face difficulties in replicating that scale. To alleviate these difficulties and to develop and share knowledge, the community needs open architecture frameworks for simulation, chip design, and software exploration that support extensibility, scalability, and configurability, alongside an established base of verification tools and supported software. In this article, we present OpenPiton, an open source framework for building scalable architecture research prototypes from one core to 500 million cores. OpenPiton is the world's first open source, general-purpose, multithreaded manycore processor, and framework. OpenPiton is highly configurable, providing a rich design space spanning a variety of hardware parameters that researchers can change. OpenPiton designs can be emulated on FPGAs, where they can run full-stack multiuser Debian Linux. OpenPiton is designed to scale to very large core fabrics, enabling researchers to measure operating system, compiler, and software scalability. The mature codebase reflects the complexity of an industrial-grade design and provides the necessary scripts to build new chips, making OpenPiton a natural choice for computer-aided design (CAD) research. OpenPiton has been validated with a 25-core chip prototype, named Piton, and is bolstered by a validation suite that has thousands of tests, providing an environment to test new hardware designs while verifying the correctness of the whole system. OpenPiton is being actively used in research both internally to Princeton and in the wider community, as well as being adopted in education, industry, and government settings.

1. INTRODUCTION

Building processors for academic research purposes can be a risky proposition. Particularly as processors have grown in size, and with the focus on multicore and manycore processors,^{17, 19, 20, 21, 14, 22, 6} the number of potential points of failure in chip fabrication has increased drastically. To combat this, the community needs well-tested, open-source, scalable frameworks that they can rely on as baselines to work from and compare against. To reduce “academic time-to-publication”, these frameworks must provide robust software tools, mature full-system software stacks, rely on industry-standard languages, and provide thorough test suites. Additionally, to support research in a broad variety of fields,

these frameworks must be highly configurable, be synthesizable to FPGA and ASIC for prototyping purposes, and provide the basis for others to tape-out (manufacture) their own, modified academic chips. Building and supporting such an infrastructure is a major undertaking which has prevented such prior designs. Our framework, OpenPiton, attacks this challenge and provides all of these features and more.

OpenPiton is the world's first open source, general-purpose, multithreaded manycore processor. OpenPiton is scalable and portable; the architecture supports addressing for up to 500-million cores, supports shared memory both within a chip and across multiple chips, and has been designed to easily enable high performance 1000+ core microprocessors and beyond. The design is implemented in **industry-standard Verilog HDL** and does not require the use of any new languages. OpenPiton enables research from the small to the large with demonstrated implementations from the slimmed-down, single-core PicoPiton, which is emulated on a \$160 Xilinx Artix 7 at 29.5MHz, up to the 25-core Piton processor which targeted a 1GHz operating point and was recently validated and thoroughly characterized.^{12, 13}

The OpenPiton platform shown in Figure 1 is a modern, tiled, manycore design consisting of a 64-bit architecture using the mature SPARC v9 ISA with P-Mesh: our scalable cache coherence protocol and network on chip (NoC). OpenPiton builds upon the industry-hardened, open-source OpenSPARC T1^{15, 1, 18} core, but sports a completely scratch-built uncore (caches, cache-coherence protocol, NoCs, NoC-based I/O bridges, etc), a new and modern simulation framework, configurable and portable FPGA scripts, a complete set of scripts enabling synthesis and implementation of ready-to-manufacture chips, and full-stack multiuser Debian Linux support. OpenPiton is available for download at <http://www.openpiton.org>.

OpenPiton has been designed as a platform to enable at-scale research. An explicit design goal of OpenPiton is that it should be easy to use by other researchers. To support this, OpenPiton provides a high degree of integration and configurability as shown in Table 1. Unlike many other designs where the pieces are provided, but it is up to the

The original version of this paper is entitled “*OpenPiton: An Open Source Manycore Research Framework*” and was published in *Proceedings of ASPLOS 2016*, Atlanta, GA, April 2–6, 2016, ACM.

Figure 1. OpenPiton Architecture. Multiple manycore chips are connected together with chipset logic and networks to build large scalable manycore systems. OpenPiton's cache coherence protocol extends off chip.

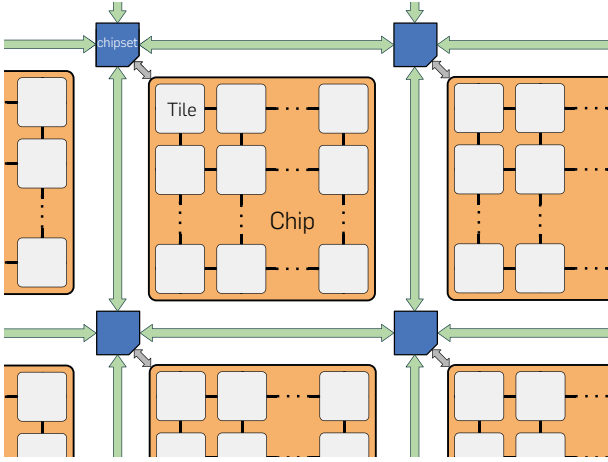


Table 1. Supported OpenPiton configuration options. Bold indicates default values. (*Associativity reduced to 2-ways at smallest size).

Component	Configurability options
Cores (per chip)	Up to 65,536
Cores (per system)	Up to 500 million
Threads per core	1/2/4
Floating-point unit	Present /Absent
Stream-processing unit	Present/ Absent
TLBs	8/ 16 /32/64 entries
L1 I-cache	8*/ 16 /32KB
L1 D-cache	4*/ 8 /16KB
L1.5 cache	Number of sets, ways (8KB, 4-way)
L2 cache (per tile)	Number of sets, ways (64KB, 4-way)
Intra-chip topologies	2D mesh , crossbar
Inter-chip topologies	2D mesh, 3D mesh, crossbar, butterfly network
Bootloading	SD/SDHC card, UART

user to compose them together, OpenPiton is designed with all of the components integrated into the same, easy-to-use, build infrastructure providing **push-button scalability**. Researchers can easily deploy OpenPiton's source code, add in modifications, and explore their novel research ideas in the setting of a fully working system. Thousands of targeted, high-coverage test cases are provided to enable researchers to innovate with a safety net that ensures functionality is maintained. OpenPiton's open source nature also makes it easy to release modifications and reproduce previous work for comparison or reuse.

Rather than simply being a platform designed by computer architects for use by computer architects, OpenPiton enables researchers in other fields including operating systems (OS), security, compilers, runtime tools, systems, and computer-aided design (CAD) tools to conduct research at-scale. In order to enable such a wide range of applications, OpenPiton is configurable and extensible. The number of cores, attached I/O, size of caches, in-core parameters, and network topology are all configurable from a command-line

option or configuration file. OpenPiton is easy to extend; the presence of a well documented core, a well documented coherence protocol, and an easy-to-interface NoC make adding research features straightforward. Research extensions to OpenPiton that have already been built include several novel memory system explorations, an Oblivious RAM controller, and a new in-core thread scheduler. The validated and mature ISA and software ecosystem support OS and compiler research. The release of OpenPiton's scripts for FPGA emulation and chip manufacture make it easy for others to port to new FPGAs or semiconductor process technologies. In particular, this enables CAD researchers who need large netlists to evaluate their algorithms at-scale.

2. THE OPENPITON PLATFORM

OpenPiton is a tiled, manycore architecture, as shown in Figure 1. It is designed to be scalable, both intra-chip and inter-chip, using the P-Mesh cache coherence system.

Intra-chip, tiles are connected via three P-Mesh networks on-chip (NoCs) in a scalable 2D mesh topology (by default). The NoC router address space supports scaling up to 256 tiles in each dimension within a single OpenPiton chip (64K cores/chip).

For inter-chip communication, the chip bridge extends the three NoCs off-chip, connecting the tile array (through the tile in the upper-left) to off-chip logic (chipset). The chipset may be implemented on an FPGA, as a standalone chip, or integrated into an OpenPiton chip.

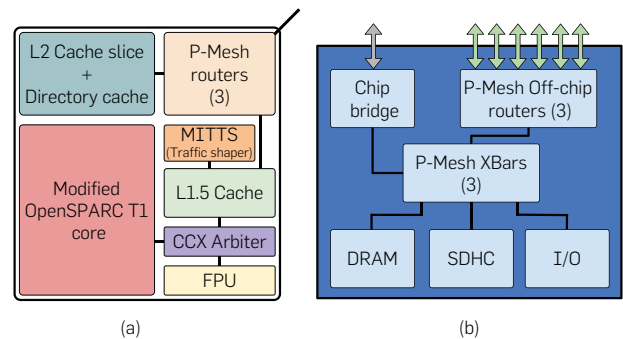
The extension of the P-Mesh NoCs off-chip allows the seamless connection of multiple OpenPiton chips to create a larger system, as shown in Figure 1. OpenPiton's cache-coherence extends off-chip as well, enabling shared-memory across multiple chips, for the study of even larger shared-memory manycore systems.

2.1. Tile

The architecture of a tile is shown in Figure 2a. A tile consists of a core, an L1.5 cache, an L2 cache, a floating-point unit (FPU), a CPU-Cache Crossbar (CCX) arbiter, a Memory Inter-arrival Time Traffic Shaper (MITTS), and three P-Mesh NoC routers.

The L2 and L1.5 caches connect directly to all three NoC routers, and all messages entering and leaving the tile traverse these interfaces. The CCX is the crossbar interface used

Figure 2. Architecture of (a) a tile and (b) chipset.



in the OpenSPARC T1 to connect the cores, L2 cache, FPU, I/O, etc.¹ In OpenPiton, the L1.5 and FPU are connected to the core by CCX.

2.2. Core

OpenPiton uses the open-source OpenSPARC T1¹⁵ core with modifications. This core was chosen because of its industry-hardened design, multi-threaded capability, simplicity, and modest silicon area requirements. Equally important, the OpenSPARC framework has a stable code base, implements a mature ISA with compiler and OS support, and comes with a large test suite.

In the default configuration for OpenPiton, as used in Piton, the number of threads is reduced from four to two and the stream processing unit (SPU) is removed from the core to save area. The default Translation Lookaside Buffer (TLB) size is 16 entries but can be increased to 32 or 64, or decreased down to 8 entries.

Additional configuration registers were added to enable extensibility within the core. They are useful for adding additional functionality to the core which can be configured from software, for example enabling/disabling functionality, configuring different modes of operation, etc.

2.3. Cache hierarchy

OpenPiton's cache hierarchy is composed of three cache levels. Each tile in OpenPiton contains private L1 and L1.5 caches and a slice of the distributed, shared L2 cache. The data path of the cache hierarchy is shown in Figure 3.

The memory subsystem maintains cache coherence using our coherence protocol, called P-Mesh. It adheres to the memory consistency model used by the OpenSPARC T1. Coherent messages between L1.5 caches and L2 caches communicate through three NoCs, carefully designed to ensure deadlock-free operation.

L1 caches. The L1 caches are reused from the OpenSPARC T1 design with extensions for configurability. They are composed of separate L1 instruction and L1 data caches, both of which are write-through and 4-way set-associative. By default, the L1 data cache is an 8KB cache and its line size is 16-bytes. The 16KB L1 instruction cache has a 32-byte line size.

L1.5 data cache. The L1.5 (comparable to L2 caches in other processors) both transduces the OpenSPARC T1's CCX protocol to P-Mesh's NoC coherence packet formats, and acts as a write-back layer, caching stores from the write-through L1 data cache. Its parameters match the L1 data cache by default.

The L1.5 communicates to and from the core through the CCX bus, preserved from the OpenSPARC T1. When a

memory request results in a miss, the L1.5 translates and forwards it to the L2 through the NoC channels. Generally, the L1.5 issues requests on NoC1, receives data on NoC2, and writes back modified cache lines on NoC3, as shown in Figure 3.

The L1.5 is inclusive of the L1 data cache; each can be independently sized with independent eviction policies. For space and performance, the L1.5 does not cache instructions—these cache lines are bypassed directly to the L2 cache.

L2 cache. The L2 cache (comparable to a last-level L3 cache in other processors) is a distributed, write-back cache shared by all tiles. The default cache configuration is 64KB per tile and 4-way set associativity, but both the cache size and associativity are configurable. The cache line size is 64 bytes, larger than the line sizes of caches lower in the hierarchy. The integrated directory cache has 64 bits per entry, so it can precisely keep track of up to 64 sharers by default.

The L2 cache is inclusive of the private caches (L1 and L1.5). Cache line way mapping between the L1.5 and the L2 is independent and is entirely subject to the replacement policy of each cache. Since the L2 is distributed, cache lines consecutively mapped in the L1.5 are likely to be distributed across multiple L2 tiles (L2 tile referring to a portion of the distributed L2 cache in a single tile).

The L2 is the point of coherence for all cacheable memory requests. All cacheable memory operations (including atomic operations such as compare-and-swap) are ordered, and the L2 strictly follows this order when servicing requests. The L2 also keeps the instruction and data caches coherent, per the OpenSPARC T1's original design. When a line is present in a core's L1 instruction cache and is loaded as data, the L2 sends invalidations to the relevant instruction caches before servicing the load.

2.4. P-Mesh network on-chip

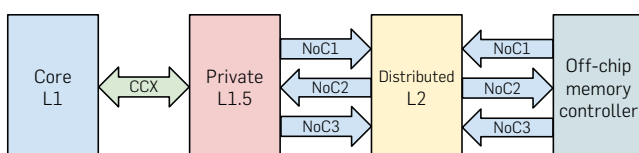
There are three P-Mesh NoCs in an OpenPiton chip. The NoCs provide communication between the tiles for cache coherence, I/O, memory traffic, and inter-core interrupts. They also route traffic destined for off-chip to the chip bridge. The packet format contains 29 bits of core addressability, making it scalable up to 500 million cores.

To ensure deadlock-free operation, the L1.5 cache, L2 cache, and memory controller give different priorities to different NoC channels; NoC3 has the highest priority, next is NoC2, and NoC1 has the lowest priority. Thus, NoC3 will never be blocked. In addition, all hardware components are designed such that consuming a high priority packet is never dependent on lower priority traffic.

Classes of coherence operations are mapped to NoCs based on the following rules, as depicted in Figure 3:

- NoC1 messages are initiated by requests from the private cache (L1.5) to the shared cache (L2).
- NoC2 messages are initiated by the shared cache (L2) to the private cache (L1.5) or memory controller.
- NoC3 messages are responses from the private cache (L1.5) or memory controller to the shared cache (L2).

Figure 3. OpenPiton's memory hierarchy datapath.



2.5. Chipset

The chipset, shown in Figure 2b, houses the I/O, DRAM controllers, chip bridge, P-Mesh chipset crossbar, and P-Mesh inter-chip network routers. The chip bridge demultiplexes traffic from the attached chip back into the three physical NoCs. The traffic then passes through a Packet Filter (not shown), which modifies packet destination addresses based on the memory address in the request and the set of devices on the chipset. The chipset crossbar (a modified network router) then routes the packets to their correct destination device. If the traffic is not destined for this chipset, it is passed to the inter-chip network routers, which route the traffic to another chipset according to the inter-chip routing protocol. Traffic destined for the attached chip is directed back through similar paths to the chip bridge.

Inter-chip routing. The inter-chip network router is configurable in terms of router degree, routing algorithm, buffer size, etc. This enables flexible exploration of different router configurations and network topologies. Currently, we have implemented and verified crossbar, 2D mesh, 3D mesh, and butterfly networks. Customized topologies can be explored by reconfiguring the network routers.

2.6. Configurability

OpenPiton was designed to be a configurable platform, making it useful for many applications. Table 1 shows OpenPiton's configurability options, highlighting the large design space that it offers.

PyHP for Verilog. In order to provide low effort configurability of our Verilog RTL, we make use of a Python pre-processor, the Python Hypertext Processor (PyHP).¹⁶ PyHP was originally designed for Python dynamic webpage generation and is akin to PHP. We have adapted it for use with Verilog code. Parameters can be passed into PyHP, and arbitrary Python code can be used to generate testbenches or modules. PyHP enables extensive configurability beyond what is possible with Verilog generate statements alone.

Core and cache configurability. OpenPiton's core configurability parameters are shown in Table 1. The default parameters are shown in bold. OpenPiton preserves the OpenSPARC T1's ability to modify TLB sizes (from 8 to 64, in powers of two), thread counts (from 1 to 4), and the presence or absence of the FPU and SPU. Additionally, OpenPiton's L1 data and instruction caches can be doubled or halved in size (associativity drops to 2 when reducing size).

Leveraging PyHP, OpenPiton provides parameterizable memories for simulation or FPGA emulation. In addition, custom or proprietary memories can easily be used for chip development. This parameterization enables the configurability of cache parameters. The size and associativity of the L1.5 and L2 caches are configurable, though the line size remains static.

Manycore scalability. PyHP also enables the creation of scalable meshes of cores, drastically reducing the code size and complexity in some areas adopted from the original OpenSPARC T1. OpenPiton automatically generates all

core instances and wires for connecting them from a single template instance. This reduces code complexity, improves readability, saves time when modifying the design, and makes the creation of large meshes straightforward. The creation of large two-dimensional mesh interconnects of up to 256×256 tiles is reduced to a single instantiation. The mesh can be any rectangular configuration, and the dimensions do not need to be powers of two. This was a necessary feature for the 5×5 (25-core) Piton processor.

NoC topology configurability. P-Mesh provides other NoC connection topologies than the default two-dimensional mesh used in OpenPiton. The coherence protocol only requires that messages are delivered in-order from one point to another point. Since there are no inter-node ordering requirements, the NoC can easily be swapped out for a crossbar, higher dimension router, or higher radix design. Our configurable P-Mesh router can be reconfigured to a number of topologies shown in Table 1. For intra-chip use, OpenPiton can be configured to use a crossbar, which has been tested with four and eight cores with no test regressions. Other NoC research prototypes can easily be integrated and their performance, energy, and other characteristics can be determined through RTL, gate-level simulation, or FPGA emulation.

Chipset configurability. The P-Mesh chipset crossbar is configurable in the number of ports to connect the myriad devices OpenPiton users may have. There is a single XML file where the chipset devices and their address ranges are specified, so connecting a new device needs only a Verilog instantiation and an XML entry. PyHP is used to automatically connect the necessary P-Mesh NoC connections and Packet Filters.

We have so far connected a variety of devices through P-Mesh on the chipset. These include DRAM, Ethernet, UART, SD, SDHC, VGA, PS/2 keyboards, and even the MIAOW open source GPU.² These devices are driven by the OpenPiton core and perform their own DMA where necessary, routed over the chipset crossbar.

Multi-chip scalability. Similar to the on-chip mesh, PyHP enables the generation of a network of chips starting with the instantiation of a single chip. OpenPiton provides an address space for up to 8192 chips, with 65,536 cores per chip. By using the scalable P-Mesh cache coherence mechanism built into OpenPiton, half-billion core systems can be built. This configurability enables the building of large systems to test ideas at scale.

3. VALIDATION

3.1. Platform stability

One of the benefits of OpenPiton is its stability, maturity, and active support. Much of this is inherited from the OpenSPARC T1 core, which has a stable code base and has been studied for years, allowing the code to be reviewed and bugs fixed by many people. In addition, it implements a mature, commercial, and open ISA, SPARC V9. This means there is existing full tool chain support for OpenPiton, including Debian Linux OS support, a compiler, and an assembler. SPARC is supported on a number of OSs including Debian Linux, Oracle's

Linux for SPARC,^a and OpenSolaris (and its successors). Porting the OpenSPARC T1 hypervisor required changes to fewer than 10 instructions, and a newer Debian Linux distribution was modified with open source, readily available, OpenSPARC T1-specific patches written as part of Lockbox.^{3,4}

OpenPiton provides additional stability on top of what is inherited from OpenSPARC T1. The tool flow was updated to modern tools and ported to modern Xilinx FPGAs. OpenPiton is also used extensively for research internal to Princeton. This means there is active support for OpenPiton, and the code is constantly being improved and optimized, with regular releases over the last several years. In addition, the open sourcing of OpenPiton has strengthened its stability as a community has built.

Validation. When designing large scale processors, simulation of the hardware design is a must. OpenPiton supports one open source and multiple commercial Verilog simulators, which can simulate the OpenPiton design at rates up to tens or hundreds of kilohertz. OpenPiton inherited and then extended the OpenSPARC T1's large test suite with thousands of directed assembly tests, randomized assembly test generators, and tests written in C. This includes tests for not only the core, but the memory system, I/O, cache coherence protocol, etc. Additionally, the extensions like Execution Drafting (ExecD) (Section 4.1.1) have their own test suites. When making research modifications to OpenPiton, the researcher can rely on an established test suite to ensure that their modifications did not introduce any regressions. In addition, the OpenPiton documentation details how to add new tests to validate modifications and extend the existing test suite. Researchers can also use our scripts to run large regressions in parallel (to tackle the slower individual execution), automatically produce pass/fail reports and coverage reports (as shown in Figure 4), and run synthesis to verify that synthesis-safe Verilog has been used. Our scripts support the widely used SLURM job scheduler and integrate with Jenkins for continuous integration testing.

3.2. FPGA prototyping

OpenPiton can also be emulated on FPGA, which provides the opportunity to prototype the design, emulated at tens of megahertz, to improve throughput when running our test suite or more complex code, such as an interactive operating system. OpenPiton is actively supported on three Xilinx FPGA platforms: Artix-7 (Digilent

Nexys Video), Kintex-7 (Digilent Genesys 2) and Virtex-7 (VC707 Evaluation Board). An external port is also maintained for the Zynq-7000 (ZC706 Evaluation Board). Figure 5 shows the area breakdown for a minimized “PicoPiton” core, implemented for an Artix-7 FPGA (Digilent Nexys 4 DDR).

OpenPiton designs have the same features as the Piton processor, validating the feasibility of that particular design (multicore functionality, etc.), and can include the chip bridge to connect multiple FPGAs via an FPGA Mezzanine Card (FMC) link. All of the FPGA prototypes feature a full system (chip plus chipset), using the same codebase as the chipset used to test the Piton processor.

OpenPiton on FPGA can load bare-metal programs over a serial port and can boot full stack multiuser Debian Linux from an SD/SDHC card. Booting Debian on the Genesys2 board running at 87.5MHz takes less than 4 minutes (and booting to a bash shell takes just one minute), compared to 45 minutes for the original OpenSPARC T1, which relied on a tethered MicroBlaze for its memory and I/O requests. This boot time improvement combined with our push-button FPGA synthesis and implementation scripts drastically increases productivity when testing operating system or hardware modifications.

3.3. The Princeton Piton Processor

The Piton processor prototype^{12, 13} was manufactured in March 2015 on IBM's 32 nm SOI process with a target clock frequency of 1GHz. It features 25 tiles in a 5×5 mesh on a $6\text{mm} \times 6\text{mm}$ (36mm^2) die. Each tile is two-way threaded and includes three research projects: ExecD,¹¹ CDR,⁸ and MITTS,²³ while an ORAM⁷ controller was included at the chip level. The Piton processor provides validation of OpenPiton as a research platform and shows that ideas can be taken from inception to silicon with OpenPiton.

With Piton, we also produced the first detailed power and energy characterization of an open source manycore design implemented in silicon.¹³ This included characterizing energy per instruction, NoC energy, voltage versus frequency scaling, thermal characterization, and memory system energy, among other properties. All of this was done in our lab, running on the Piton processor with the OpenPiton chipset implemented on FPGA. Performing such a characterization yielded new insights into the balance between recomputation and data movement,

a Linux for SPARC is hosted at <https://oss.oracle.com/projects/linux-sparc/>

Figure 4. Test suite coverage results by module (default OpenPiton configuration).

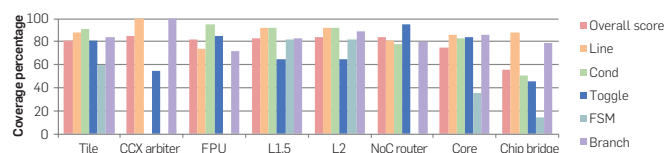
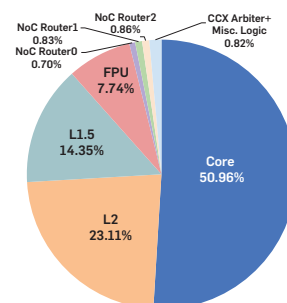
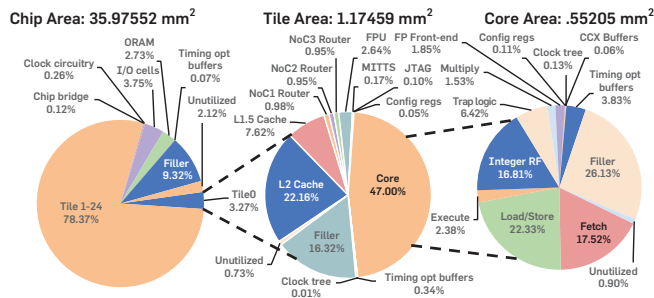


Figure 5. Tile area breakdown for FPGA PicoPiton.



the energy cost of differing operand values, and a confirmation of earlier results⁹ that showed that NoCs do not dominate manycore processors' power consumption. Our study also produced what we believe is the most detailed area breakdown of an open source manycore, which we reproduce in Figure 6. All characterization data from our study, as well as designs for the chip printed circuit board (PCB), are now open source at <http://www.openpiton.org>.

Figure 6. Detailed area breakdown of Piton at chip, tile, and core levels. Reproduced from McKeown et al.¹³



3.4. Synthesis and back-end support

OpenPiton provides scripts to aid in synthesis and back-end physical design for generating realistic area results or for manufacturing new chips based on OpenPiton. The scripts are identical to the ones used to tape-out the Piton processor, however the scripts have been made process agnostic and references to the specific technology used have been removed due to proprietary foundry intellectual property concerns. Directions are included with OpenPiton which describe how to port to a new foundry kit. This allows the user to download OpenPiton, link to the necessary process development kit files, and run our full tool flow to produce the chip layout for a new instance of OpenPiton. In this sense, OpenPiton is portable across process technologies and provides a complete ecosystem to implement, test, prototype, and tape-out (manufacture) research chips.

4. APPLICATIONS

Table 2 presents a taxonomy of open source processors which highlights important parameters for research. Since OpenPiton's first release in 2015, it has been used across a wide range of applications and research domains, some of which are described in this section.

Table 2. Taxonomy of differences of open source processors (table data last checked in April 2018).

Processor	Architecture	FPU	OS	MMU	HW multi-threaded	Multicore/ manycore/ GPU	Prototype core count	NoC	HDL	Back-end scripts	License
pAVR	8b AVR	×	×	×	×	No	—	×	VHDL	×	GPL v2
openMSP430	16bMSP430	×	✓	×	×	No	—	×	Verilog	×	BSD
CPU86	16b x86	×	✓	×	×	No	—	×	VHDL	×	GPL
Zet	16b x86	×	✓	×	×	No	—	×	Verilog	×	GPL v3
LatticeMico32	32b LatticeMico32	×	✓	×	×	No	—	×	Verilog	×	GPL
ZPU	32b MIPS	×	✓	×	×	No	—	×	VHDL	×	FreeBSD & GPL
SecretBlaze	32b MicroBlaze	×	×	×	×	No	—	×	VHDL	×	GPL v3
AltOr32	32b ORBIS	×	✓	×	✓	No	—	×	Verilog	×	LGPL v3
aeMB	32b MicroBlaze	×	✓	×	×	No	—	×	Verilog	×	LGPL v3
Amber	32b ARM v2a	×	✓	×	×	No	—	×	Verilog	×	LGPL
OpenRISC	32b/64b ORBIS	✓	✓	✓	×	No	—	×	Verilog	×	LGPL
MIPS32 r1	32b MIPS32 r1	×	✓	×	✓	No	—	×	Verilog	×	LGPL v3
LEON 3	32b SPARC V8	✓(\$)	✓	✓	×	SMP/AMP	—	×	VHDL	×	GPL
OpenScale	32b MicroBlaze	×	✓	×	×	Manycore	FPGA/6	✓	VHDL	×	GPL v3
XUM	32b MIPS32 r2	×	✓	×	✓	Manycore	FPGA/8	✓	Verilog	×	LGPL v3
PicoRV32	32b RISC-V	×	×	×	×	No	FPGA/1	×	Verilog	×	ISC
PULP-RI5CY	32b RISC-V	✓	✓	×	×	Manycore	Chip/9	×	SystemVerilog	×	Solderpad 0.51
PULP-Zeroriscy	32b RISC-V	×	✓	×	×	Multicore	Chip/1	×	SystemVerilog	×	Solderpad 0.51
Nyuzi GPGPU	Nyami ISA	✓	✓	✓	✓	GPGPU	FPGA	✓	SystemVerilog	×	Apache 2.0
MIAOW GPGPU	AMD Southern Islands	✓	×	×	✓	GPU	FPGA/1	✓	Verilog	×	BSD 3-Clause
OpTiMSoC	32b/64b ORBIS	✓	✓	✓	×	Manycore	FPGA/4	✓	SystemVerilog	×	MIT
Simply RISC S1	64b SPARC V9	✓	✓	✓	×	No	—	×	Verilog	×	GPL v2
BERI	64b MIPS/CHERI	✓	✓	✓	✓(BERI2)	Multicore	FPGA/4	×	Bluespec	×	BERI HW-SW
OpenSPARC T1/T2	64b SPARC V9	✓	✓	✓	✓	Multicore	Chip/8	×	Verilog	×	GPL v2
Rocket	64b RISC-V	✓	✓	✓	✓	Manycore	Chip/8	✓	Chisel	×	BSD 3-Clause
AnyCore	64b RISC-V	×	×	×	×	No	Chip/1	×	SystemVerilog	×	BSD 3-Clause
PULP-Ariane	64b RISC-V	×	✓	×	×	Manycore	Chip/1	×	SystemVerilog	×	Solderpad 0.51
BOOM	64b RISC-V	✓	✓	✓	×	Manycore	FPGA	✓	Chisel	×	BSD 3-Clause
OpenPiton	64b SPARC V9	✓	✓	✓	✓	Manycore	Chip/25	✓	Verilog	✓	BSD 3-Clause & GPL v2

* (RTOS)

4.1. Internal research case studies

Execution Drafting. Execution Drafting¹¹ (ExecD) is an energy saving microarchitectural technique for multi-threaded processors, which leverages duplicate computation. ExecD takes over the thread selection decision from the OpenSPARC T1 thread selection policy and instruments the front-end to achieve energy savings. ExecD required modifications to the OpenSPARC T1 core and thus was not as simple as plugging a standalone module into the OpenPiton system. The core microarchitecture needed to be understood, and the implementation tightly integrated with the core. Implementing ExecD in OpenPiton revealed several implementation details that had been abstracted away in simulation, such as tricky divergence conditions in the thread synchronization mechanisms. This reiterates the importance of taking research designs to implementation in an infrastructure like OpenPiton.

ExecD must be enabled by an ExecD-aware operating system. Our public Linux kernel and OpenPiton hypervisor repositories contain patches intended to add support for ExecD. These patches were developed as part of a single-semester undergraduate OS research project.

Coherence Domain Restriction. Coherence Domain Restriction⁸ (CDR) is a novel cache coherence framework designed to enable large scale shared memory with low storage and energy overhead. CDR restricts cache coherence of an application or page to a subset of cores, rather than keeping global coherence over potentially millions of cores. In order to implement it in OpenPiton, the TLB is extended with extra fields and both the L1.5 and L2 cache are modified to fit CDR into the existing cache coherence protocol. CDR is specifically designed for large scale shared memory systems such as OpenPiton. In fact, OpenPiton's million-core scalability is not feasible without CDR because of increasing directory storage overhead.

Memory Inter-arrival Time Traffic Shaper. The Memory Inter-arrival Time Traffic Shaper²³ (MITTS) enables a manycore system or an IaaS cloud system to provision memory bandwidth in the form of a memory request interarrival time distribution at a per-core or per-application basis. A runtime system configures MITTS knobs in order to optimize different metrics (e.g., throughput, fairness). MITTS sits at the egress of the L1.5 cache, monitoring the memory requests and stalling the L1.5 when it uses bandwidth outside its allocated distribution. MITTS has been integrated with OpenPiton and works on a per-core granularity, though it could be easily modified to operate per-thread.

MITTS must also be supported by the OS. Our public Linux kernel and OpenPiton hypervisor repositories contain patches for supporting the MITTS hardware. With these patches, developed as an undergraduate thesis project, Linux processes can be assigned memory inter-arrival time distributions, as they would in an IaaS environment where the customer paid for a particular distribution corresponding with their application's behavior. The OS configures the MITTS bins to correspond with each process's

allocated distribution, and MITTS enforces the distribution accordingly.

4.2. External research use

A number of external researchers have already made considerable use of OpenPiton. In a CAD context, Lerner et al.¹⁰ present a development workflow for improving processor lifetime, based on OpenPiton and the gem5 simulator, which is able to improve the design's reliability time by 4.1×.

OpenPiton has also been used in a security context as a testbed for hardware trojan detection. OpenPiton's FPGA emulation enabled Elnaggar et al.⁵ to boot full-stack Debian Linux and extract performance counter information while running SPEC benchmarks. This project moved quickly from adopting OpenPiton to an accepted publication in a matter of months, thanks in part to the full-stack OpenPiton system that can be emulated on FPGA.

Oblivious RAM (ORAM)⁷ is a memory controller designed to eliminate memory side channels. An ORAM controller was integrated into the 25-core Piton processor, providing the opportunity for secure access to off-chip DRAM. The controller was directly connected to OpenPiton's NoC, making the integration straightforward. It only required a handful of files to wrap an existing ORAM implementation, and once it was connected, its integration was verified in simulation using the OpenPiton test suite.

4.3. Educational use

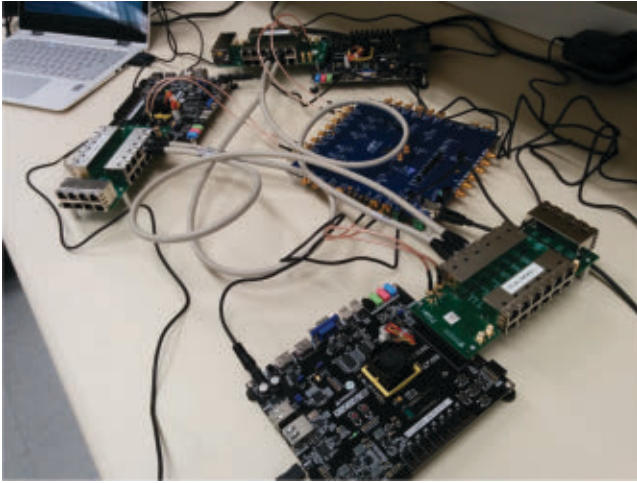
We have been using OpenPiton in coursework at Princeton, in particular our senior undergraduate Computer Architecture and graduate Parallel Computation classes. A few of the resulting student projects are described here.

Core replacement. Internally, we have tested replacements for the OpenSPARC T1 core with two other open source cores. These modifications replaced the CCX interface to the L1.5 cache with shims which translate to the L1.5's interface signals. These shims require very little logic but provide the cores with fully cache-coherent memory access through P-Mesh. We are using these cores to investigate manycore processors with heterogeneous ISAs.

Multichip network topology exploration. A senior undergraduate thesis project investigated the impact of interchip network topologies for large manycore processors. Figure 7 shows multiple FPGAs connected over a high-speed serial interface, carrying standard P-Mesh packets at 9 gigabits per second. The student developed a configurable P-Mesh router for this project which is now integrated as a standard OpenPiton component.

MIAOW. A student project integrated the MIAOW open source GPU² with OpenPiton. An OpenPiton core and a MIAOW core can both fit onto a VC707 FPGA with the OpenPiton core acting as a host, in place of the Microblaze that was used in the original MIAOW release. The students added MIAOW to the chipset crossbar with a single entry in its XML configuration. Once they implemented a native P-Mesh interface to replace the original AXI-Lite interface, MIAOW could

Figure 7. Three OpenPiton FPGAs connected by 9 gigabit per second serial P-Mesh links.



directly access its data and instructions from memory without the core's assistance.

Hardware transactional memory. Another student project was the implementation of a hardware transactional memory system in OpenPiton. The students learned about the P-Mesh cache coherence protocol from the OpenPiton documentation, before modifying it, including adding extra states to the L1.5 cache, and producing a highly functional prototype in only six weeks. The OpenPiton test suite was central to verifying that existing functionality was maintained in the process.

Cache replacement policies. A number of student groups have modified the cache replacement policies of both the L1.5 and L2 caches. OpenPiton enabled them to investigate the performance and area tradeoffs of their replacement policies across multiple cache sizes and associativities in the context of a full-stack system, capable of running complex applications.

4.4. Industrial and governmental use

So far we are aware of multiple CAD vendors making use of OpenPiton internally for testing and educational purposes. These users provide extra confidence that the RTL written for OpenPiton will be well supported by industrial CAD tools, as vendors often lack large scale designs to validate the functionality of their tools. In government use, DARPA has identified OpenPiton as a benchmark for use in the POSH program.

5. FUTURE

OpenPiton has a bright future. It not only has active support from researchers at Princeton but has a vibrant external user base and development community. The OpenPiton team has run four tutorials at major conferences and numerous tutorials at interested universities and will continue to run more tutorials. The future roadmap for OpenPiton includes adding additional configurability, support for more FPGA platforms and vendors, the ability to emulate in the cloud

by using Amazon AWS F1 instances, more core types plugged into the OpenPiton infrastructure, and integration with other emerging open source hardware projects. OpenPiton has demonstrated the ability to enable research at hardware speeds, at scale, and across different areas of computing research. OpenPiton and other emerging open source hardware projects have the potential to have significant impact not only on how we conduct research and educate students, but also design chips for commercial and governmental applications.

Acknowledgments

This material is based on research sponsored by the NSF under Grants No. CNS-1823222, CCF-1217553, CCF-1453112, and CCF-1438980, AFOSR under Grant No. FA9550-14-1-0148, Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement No. FA8650-18-2-7846 and FA8650-18-2-7852 and DARPA under Grants No. N66001-14-1-4040 and HR0011-13-2-0005. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA), the NSF, AFOSR, DARPA, or the U.S. Government. We thank Paul Jackson, Ting-Jung Chang, Ang Li, Fei Gao, Katie Lim, Felix Madutsa, and Kathleen Feng for their important contributions to OpenPiton. C

References

1. *OpenSPARC T1 Microarchitecture Specification*. Santa Clara, CA, 2006.
2. Balasubramanian, R., Gangadhar, V., Guo, Z., Ho, C.-H., Joseph, C., Menon, J., Drummond, M.P., Paul, R., Prasad, S., Valathol, P., Sankaralingam, K. Enabling GPGPU low-level hardware explorations with MIAOW: An open-source RTL implementation of a GPGPU. *ACM Trans. Archit. Code Optim.* 12, 2 (June 2015).
3. Bittman, D., Capelis, D., Long, D. Introducing SeaOS. In *2014 International Conference on Information Science and Applications (ICISA)*, May 2014, 1–3.
4. Capelis, D.J. *Lockbox: Helping Computers Keep Your Secrets*. Technical Report UCSC-WASP-15-02, University of California, Santa Cruz, Nov. 2015.
5. Elnaggar, R., Chakrabarty, K., Tahoori, M.B. Run-time hardware trojan detection using performance counters. In *2017 IEEE International Test Conference (ITC)*, Oct. 2017, 1–10.
6. Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., Burger, D. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture, ISCA '11* (New York, NY, USA, 2011), ACM, 365–376.
7. Fletcher, C.W., Ren, L., Kwon, A., van Dijk, M., Devadas, S. Freecursive ORAM: [nearly] free recursion and integrity verification for position-based oblivious ram. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '15* (New York, NY, USA, 2015), ACM, 103–116.
8. Fu, Y., Nguyen, T.M., Wentzlaff, D. Coherence domain restriction on large scale systems. In *Proceedings of the 48th International Symposium on Microarchitecture, MICRO-48* (New York, NY, USA, 2015), ACM, 686–698.
9. Kim, J.S., Taylor, M.B., Miller, J., Wentzlaff, D. Energy characterization of a tiled architecture processor with on-chip networks. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ISLPED '03* (New York, NY, USA, 2003), ACM, 4–427.
10. Lerner, S., Taskin, B. Workload-aware ASIC flow for lifetime improvement of multi-core IoT processors. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*, March 2017, 379–384.
11. McKeown, M., Balkind, J., Wentzlaff, D. Execution drafting: Energy efficiency through computation deduplication. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec. 2014, 432–444.

12. McKeown, M., Fu, Y., Nguyen, T., Zhou, Y., Balkind, J., Lavrov, A., Shahrad, M., Payne, S., Wentzlaff, D. Piton: A manycore processor for multitenant clouds. *IEEE Micro* 37, (2) (Mar. 2017), 70–80.
13. McKeown, M., Lavrov, A., Shahrad, M., Jackson, P., Fu, Y., Balkind, J., Nguyen, T., Lim, K., Zhou, Y., Wentzlaff, D. Power and energy characterization of an open source 25-core manycore processor. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018.
14. Miller, B., Brasili, D., Kiszely, T., Kuhn, R., Mehrotra, R., Salvi, M., Kulkarni, M., Varadharajan, A., Yin, S.-H., Lin, W., Hughes, A., Stysiac, B., Kandadi, V., Pragasathi, I., Hartman, D., Carlson, D., Yalala, V., Xanthopoulos, T., Meninger, S., Crain, E., Spaeth, M., Aina, A., Balasubramanian, S., Vulih, J., Tiwary, P., Lin, D., Kessler, R., Fishbein, B., Jain, A. A 32-core RISC microprocessor with network accelerators, power management and testability features. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2012, 58–60.
15. Oracle. OpenSPARC T1. <http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-1444609.html>.
16. PyHP. PyHP Official Home Page. <http://pyhp.sourceforge.net>.
17. Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., Hanrahan, P. Larrabee: A many-core $\times 86$ architecture for visual computing. *ACM Trans. Graph.* 27, (3) (Aug. 2008), 18:1–18:15.
18. Szefer, J., Zhang, W., Chen, Y.-Y., Champagne, D., Chan, K., Li, W., Cheung, R., Lee, R. Rapid single-chip secure processor prototyping on the OpenSPARC FPGA platform. In *2011 22nd IEEE International Symposium on Rapid System Prototyping (RSP)*, May 2011, 38–44.
19. Vangal, S.R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S., et al. An 80-tile sub-100-w teraops processor in 65-nm CMOS. *IEEE J. Solid-State Circuits* 43, 1 (2008), 29–41.
20. Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C.-C., Brown III, J.F., Agarwal, A. On-chip interconnection architecture of the Tile Processor. *IEEE Micro* 27, (5) (Sept. 2007), 15–31.
21. Wentzlaff, D., Jackson, C.J., Griffin, P., Agarwal, A. Configurable fine-grain protection for multicore processor virtualization. In *Proceedings of the Annual International Symposium on Computer Architecture* (Washington, DC, USA, 2012), 464–475.
22. Woo, D.H., Lee, H.-H.S. Extending Amdahl's law for energy-efficient computing in the many-core era. *Computer* 12 (2008), 24–31.
23. Zhou, Y., Wentzlaff, D. Mitts: Memory inter-arrival time traffic shaping. In *Proceedings of the 43rd International Symposium on Computer Architecture, ISCA '16* (Piscataway, NJ, USA, 2016), IEEE Press, 532–544.

Jonathan Balkind, Michael McKeown, Tri Nguyen, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, and David Wentzlaff ([jbalkind, mmckeown, trin, alavrov, mshahrad, adif, wentzlaff]@princeton.edu), Princeton University, Princeton, NJ, USA.

Yaosheng Fu* and Samuel Payne* ([yfu, spayne]@nvidia.com), NVIDIA, Santa Clara, CA, USA.

*Work was done at Princeton University.

Yanqi Zhou* (zhouyanqi@baidu.com), Baidu SVAIL, Sunnyvale, CA, USA.

Xiaohua Liang* (xialian@microsoft.com), Microsoft, Redmond, WA, USA.

Matthew Matti* (mmatti@eecs.berkeley.edu), University of California, Berkeley, USA.

Copyright held by authors/owners. Publication rights licensed to ACM.



Advertise with ACM!

Reach the innovators and thought leaders working at the cutting edge of computing and information technology through ACM's magazines, websites and newsletters.

Request a media kit with specifications and pricing:



Ilia Rodriguez
+1 212-626-0686
acmm mediasales@acm.org



Department of Electrical and Computer Engineering Graduate School of Engineering and Management Air Force Institute of Technology (AFIT) Dayton, Ohio Faculty Position

The Department of Electrical and Computer Engineering at the Air Force Institute of Technology is seeking applications for a tenured or tenure-track faculty position. All academic ranks will be considered. Applicants must have an earned doctorate in Electrical Engineering, Computer Engineering, Computer Science, or a closely affiliated discipline by the time of their appointment (anticipated 1 September 2020).

We are particularly interested in applicants specializing in one or more of the following areas: autonomy, artificial intelligence / machine learning, navigation with or without GPS, cyber security, and VLSI. Candidates in other areas of specialization are also encouraged to apply. This position requires teaching at the graduate level as well as establishing and sustaining a strong DoD relevant externally funded research program with a sustainable record of related peer-reviewed publications.

The Air Force Institute of Technology (AFIT) is the premier Department of Defense (DoD) institution for graduate education in science, technology, engineering, and management, and has a Carnegie Classification as a High Research Activity Doctoral University. The Department of Electrical and Computer Engineering offers accredited M.S. and Ph.D. degree programs in Electrical Engineering, Computer Engineering, and Computer Science as well as an MS degree program in Cyber Operations.

Applicants must be U.S. citizens. Full details on the position, the department, applicant qualifications, and application procedures can be found at <http://www.afit.edu/ENG/>. Review of applications will begin on January 6, 2020. The United States Air Force is an equal opportunity, affirmative action employer.

CAREERS

Boston College

Non Tenure-Track Positions in Computer Science

The Computer Science Department of Boston College seeks to fill one or more non-tenure-track teaching positions, as well as shorter-term visiting teaching positions. All applicants should be committed to excellence in undergraduate education and be able to teach a broad variety of undergraduate computer science courses. Faculty in longer-term positions will also participate in the development of new courses that reflect the evolving landscape of the discipline.

Minimum requirements for the title of Assistant Professor of the Practice, and for the title of Visiting Assistant Professor, include a Ph.D. in Computer Science or closely related discipline.

Candidates without a Ph.D. would be eligible for the title of Lecturer or Visiting Lecturer.

We will begin reviewing applications on October 15, 2019 and will continue considering applications until the positions are filled. Applicants should submit a cover letter, CV, and a separate teaching statement and arrange for three confidential letters of recommendation that comment on their teaching performance to be uploaded directly to Interfolio. To apply go to <https://apply.interfolio.com/68339>. Boston College conducts background checks as part of the hiring process. Information about the university and our department is available at <https://www.bc.edu> and <http://cs.bc.edu>.

Boston College is a Jesuit, Catholic university that strives to integrate research excellence with a foundational commitment to formative liberal arts education. We encourage applications from candidates who are committed to fostering a diverse and inclusive academic community. Boston College is an affirmative action/equal opportunity employer.

Boston College

Tenure-Track Assistant Professor of Computer Science

The Computer Science Department of Boston College seeks a tenure-track Assistant Professor beginning in the 2020-2021 academic year. Successful candidates for the position will be expected to develop strong research programs that can attract external funding in an environment that also values high-quality undergraduate teaching. Outstanding candidates in all areas of Computer Science will be considered, with a preference for those who demonstrate a potential to contribute to cross-disciplinary teaching and research in conjunction with the planned Schiller Institute for Integrated Science and Society at Boston College.

A Ph.D. in Computer Science or a closely related discipline is required. See <http://cs.bc.edu> and <https://www.bc.edu/bc-web/schools/mcas/sites/schiller-institute.html> for more information. Application review is ongoing. Boston Col-

lege conducts background checks as part of the hiring process.

Submit a cover letter, a detailed CV and teaching and research statements. Arrange for three confidential letters of recommendation to be uploaded directly to Interfolio. To apply go to <https://apply.interfolio.com/68273>.

Boston College is a Jesuit, Catholic university that strives to integrate research excellence with a foundational commitment to formative liberal arts education. We encourage applications from candidates who are committed to fostering a diverse and inclusive academic community. Boston College is an affirmative action/equal opportunity employer.

Florida International University

Computing and Information Sciences (SCIS) Open-Rank Tenure Track/Tenured Positions

FIU's School of Computing and Information Sciences (SCIS) is a rapidly growing program of excellence at Florida International University (FIU). The School has 30 tenure-track faculty members and over 3,000 students, including over 90 Ph.D. students. The School is engaged in on-going and exciting new and expanding programs for research, education and outreach. The School offers B.S., M.S., and Ph.D. degrees in Computer Science, and M.S. degrees in Telecommunications and Networking, Cyber-security, Data Science, and Information Technology as well as B.S./B.A. degrees in Information Technology. NSF ranks FIU 39th nationwide in externally-funded research expenditures. SCIS has six research centers/clusters with first-class computing and support infrastructure, and enjoys broad and dynamic industry and international partnerships.

The School of Computing and Information Sciences invites applications from exceptionally qualified faculty at all levels with particular emphasis in cybersecurity, data science, AI, and machine learning. Ideal candidates for junior positions should have a record of exceptional research in their early careers and a demonstrated ability to pursue and lead a research program. Candidates for senior positions must have an active and sustainable record of excellence in funded research, publications and professional service as well as demonstrated leadership in collaborative or interdisciplinary research. In addition to developing or expanding a high-quality research program, all successful applicants must be committed to excellence in teaching at both the graduate and undergraduate levels.

Applications are encouraged from candidates with highly transformative research programs and seminal ideas that extend the frontiers of computing and networking across other disciplines. A Ph.D. in Computer Science or related disciplines is required.

Qualified candidates for Open-Rank Tenure-Track/Tenured faculty positions are encouraged to apply to 519661. Visit [\[fiu.edu/\]\(http://fiu.edu/\) and attach cover letter, curriculum vitae, statement of teaching philosophy, research statement, etc. as individual attachments. Candidates are required to provide names and contact information for at least three references who will be contacted as determined by the search committee. To receive full consideration, applications and required materials should be received by November 10th, 2019. Review will continue until the position has been filled.](https://facultycareers.</p></div><div data-bbox=)

FIU is a member of the State University System of Florida and an Equal Opportunity, Equal Access Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, disability status, protected veteran status, or any other characteristic protected by law.

Florida State University

Department of Computer Science

Tenure-Track Assistant Professor Position in Data Sciences

The Department of Computer Science at the Florida State University invites applications for a tenure-track Assistant Professor position to begin August 2020. The position is a 9-month, full-time, tenure-track, and benefits eligible. We are seeking outstanding theoretical and applied applicants in the broad area of Data Sciences. The focused areas include Data Analytics, Visualization, Machine Learning, Computer Architecture, Human Computer Interaction, and Algorithms.

Applicants should hold a Doctoral degree from an accredited institution or the highest degree appropriate in the field of Computer Science or closely related field at the time of appointment and have a demonstrated record of research and teaching accomplishments. A minimum of 2 years of teaching and research experience is preferred.

The department currently has 24 tenure-track and 8 specialized faculty members and offers degrees at the BS, MS, and PhD levels. Our annual research expenditure has been growing in the past several years and reached \$3.9 Million in the 2019 fiscal year. The department is an NSA/DHS Center of Academic Excellence in Cyber Defense Education (CAE/CDE) and Research (CAE-R). FSU is classified as a Carnegie Research I university. Its primary role is to serve as a center for advanced graduate and professional studies while emphasizing research and providing excellence in undergraduate education. Further information can be found at: <http://www.cs.fsu.edu>.

Screening will begin December 1, 2019. The deadline for applications is May 1, 2020. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of three references at: www.jobs.fsu.edu, select "Browse Job Openings," and search for job 46508. Questions can be e-mailed to Prof. Weikuan Yu, Faculty Search Committee Chair, recruitment@cs.fsu.edu.

The research institute for Cyber Defence (CODE) at the Bundeswehr University Munich has been substantially expanded to become the research institute for "Cyber Defence and Smart Data" of the Bundeswehr and the Federal Government institutions. CODE was established in 2013 with the objective of bringing together experts from different faculties and scientific disciplines as well as expertise from industry and government agencies to conduct research in cyber and information space. CODE pursues a comprehensive, integrated and interdisciplinary approach to implementing technical innovations and concepts for the protection of data, software and ICT infrastructures in accordance with legal and commercial framework conditions. It has already established important strategic partnerships in this area. The objective of the expansion is to unite the Bundeswehr's and the Federal Government's research initiatives in the area of Cyber Defence and Smart Data and to establish the CODE research institute as the primary point of contact in the cyber and information domains of the Bundeswehr and the Federal Government.

Research and teaching in the area of cyber security is already being carried out as part of the bachelor's and master's programs in the Department of Computer Science. A new independent master's program in Cyber Security was launched on January 1st 2018.

As part of the expansion of CODE, the Department of Computer Science at Bundeswehr University Munich seeks to appoint, at the earliest possible date:

Four Full Professorships (W3) in Information Security

The department particularly seeks to recruit in the areas of 1) **cryptology** (e.g. post-quantum, key management and protocols, cryptanalysis - including quantum computing-supported, mathematical foundations), 2) **cyber-physical system security** (e.g. distributed embedded systems, Internet of Things, real time systems, side channels, hardware security), 3) **privacy** (e.g. privacy enhancing technologies, differential privacy, k-anonymity/l-diversity, Internet anonymity) and 4) **open source intelligence** (big data, data mining, visualization, metadata, analysis of social bots).

For each professorship, the Bundeswehr University Munich is looking for a professor who, in addition to having outstanding scientific qualifications, will also contribute actively to the CODE research institute. Besides excellent research work, the new professor is expected to develop demanding courses for the master's program in Cyber Security and to provide excellent teaching in her or his respective specialist area. The applicant is also expected to carry out teaching in the bachelor's programs in computer science, and to cooperate closely in research with other departments at Bundeswehr University Munich.

As the first German IBM Q-Hub, CODE has access to an IBM quantum computer. Faculty members at Bundeswehr University are eligible to apply for funding via the usual routes, such as DFG, EU, BMBF, but in addition are able to collaborate with several federal institutions.

Courses at the Bundeswehr University Munich are taught in small groups. The professorship will be provided with superbly equipped laboratories housed in a new building that will be built in the near future. The candidate must have an excellent scientific track record, as demonstrated by a habilitation or equivalent scientific achievement, as well as relevant publications in academic journals and conferences. Proven teaching experience in her or his respective specialist area is highly desired. The new professor should have an international perspective, e.g. based on participation in international research projects, as well as experience in acquiring third-party funding. The duties will also include participation in the academic self-administration of the university. Further, the candidate will be expected to assume a gender equality based leadership role.

The Bundeswehr University Munich offers academic programs directed primarily at officer candidates and officers, who can obtain bachelor's and master's degrees in a trimester system. Study programs are complemented by an integrated program entitled "studium plus", which consists of interdisciplinary seminars, tutorials and training in key professional qualifications.

The recruitment requirements and the employment status of professors are governed by the Federal Civil Service Act (Bundesbeamtengesetz). To become an appointed civil servant (Beamter), candidates must be no older than 50 on the date of their appointment.

The university seeks to increase the number of female professors and thus explicitly invites women to submit applications. Severely disabled candidates with equal qualifications will receive preferential consideration.

Please direct any questions about the process and the offered positions to Ms. Stefanie Molnar (+49-89-6004-2634, stefanie.molnar@unibw.de).

Please submit your application documents, marked as Confidential Personnel Matter, to the **Dean of the Department of Computer Science, Professor Dr. Oliver Rose, Bundeswehr University Munich, D-85577 Neubiberg, or via email to dekanat.inf@unibw.de, by 07th January 2020.**



FSU is an Equal Opportunity/Access/Affirmative Action/Pro Disabled & Veteran Employer. FSU's Equal Opportunity Statement can be viewed at: http://www.hr.fsu.edu/PDF/Publications/diversity/EEO_Statement.pdf.

Individuals from traditionally underrepresented groups are encouraged to apply.

Florida State University
Department of Computer Science
Tenure-Track Assistant Professor Position in
Trustworthy Computing

The Department of Computer Science at the Florida State University invites applications for a tenure-track Assistant Professor position to begin August 2020. The position is a 9-month, full-time, tenure-track, and benefits eligible. We are seeking outstanding theoretical and applied applicants in the broad area of Trustworthy Computing. The focused areas include Embedded and Cyber-Physical Systems, Visualization, Computer Architecture, Human Computer Interaction, and Algorithms.

Applicants should hold a Doctoral degree from an accredited institution or the highest degree appropriate in the field of Computer Science or closely related field at the time of appointment and have a demonstrated record of research and teaching accomplishments. A minimum of 2 years of teaching and research experience is preferred.

The department currently has 24 tenure-track and 8 specialized faculty members and offers degrees at the BS, MS, and PhD levels. Our annual research expenditure has been growing in the

past several years and reached \$3.9 Million in the 2019 fiscal year. The department is an NSA/DHS Center of Academic Excellence in Cyber Defense Education (CAE/CDE) and Research (CAE-R). FSU is classified as a Carnegie Research I university. Its primary role is to serve as a center for advanced graduate and professional studies while emphasizing research and providing excellence in undergraduate education. Further information can be found at: <http://www.cs.fsu.edu>.

Screening will begin December 1, 2019. The deadline for applications is May 1, 2020. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of three references at: www.jobs.fsu.edu, select "Browse Job Openings," and search for job 46510. Questions can be e-mailed to Prof. Weikuan Yu, Faculty Search Committee Chair, recruitment@cs.fsu.edu.

FSU is an Equal Opportunity/Access/Affirmative Action/Pro Disabled & Veteran Employer. FSU's Equal Opportunity Statement can be viewed at: http://www.hr.fsu.edu/PDF/Publications/diversity/EEO_Statement.pdf.

Individuals from traditionally underrepresented groups are encouraged to apply.

Georgia Institute of Technology
School of Computational Science and
Engineering (CSE)
CSE Tenure-Track Faculty Positions

The School of Computational Science and Engineering (CSE) in the College of Computing at

the Georgia Institute of Technology invites applications for multiple openings at the Assistant Professor level (tenure-track); exceptional candidates at the Associate Professor and Professor level also will be considered. CSE focuses on foundational research of an interdisciplinary nature that enables advances in science, engineering, medical, and social domains. Applicants are expected to develop and sustain a research program in one or more of our core areas: high-performance computing, scientific and numerical computing, modeling and simulation, discrete algorithms, and large-scale data analytics (including machine learning and artificial intelligence). Applicants are expected to engage in substantive research with collaborators in other disciplines. For example, current faculty have domain expertise and/or collaborations in computational chemistry; earth sciences; biomedical and health; urban systems and smart cities; social good and sustainable development; materials and manufacturing; cybersecurity; and others. Applicants must have an outstanding record of research and a commitment to teaching.

Georgia Tech is organized into six Colleges. The School of Computational Science and Engineering resides in the College of Computing along with the School of Computer Science and the School of Interactive Computing. Joint appointments with other Schools in the College of Computing as well as Schools in other Colleges will be considered.

Applications should be submitted online through: <https://academicjobsonline.org/ajo/jobs/14887>.



University of Missouri

Assistant, Associate, Full Professors In Cyber Security & High Assurance Computing

The Department of Electrical Engineering and Computer Science at the University of Missouri seeks applications for five tenure-track/tenured positions at the rank of Assistant, Associate, and Full Professor, starting Fall 2020. These five positions are part of a University of Missouri strategic initiative in Cyber Security and complement recent hiring in that area. Applicants must have a Ph.D. in Computer Science, Computer Engineering or a closely related field. Preferred candidates will have success in research, and a strong commitment to excellence in teaching. For a candidate seeking a position at the level of Associate or Full Professor, a record of attracting external research funding appropriate to their rank is an essential factor. We also encourage applicants with a history of, or an interest in, interdisciplinary research. The focus of these hires will be in Cyber Security, broadly construed, including formal methods, systems, and theory.

We are especially interested in: Hardware and Embedded Systems Security, Secure Software Engineering (including static binary analysis, language-based security, and software verification), Wireless Security, Cloud Security, Cyber-Physical Systems and Internet of Things.

Applications will be reviewed immediately upon receipt and will continue until the positions are filled.

Application: Applicants should submit a CV, a research plan, a teaching statement and a list of three to five professional references electronically to <http://hrs.missouri.edu/find-a-job/academic> refer to Job ID 31546.

Inquiries can be directed to the search committee at securityhiring@missouri.edu.

The University of Missouri is a Tier I research institution and one of only 60 public and private U.S. universities invited to membership in the prestigious Association of American Universities. The university was founded in 1839 in Columbia as the first public university west of the Mississippi River.

MU specifically invites and encourages applications from qualified women and members of groups underrepresented in science. Equal Opportunity/Affirmative Action/ADA employer firmly committed to fostering ethnic, racial, and gender diversity in our faculty.



**Australian
National
University**

OPEN RANK TENURE TRACK – MULTIPLE COMPUTER SCIENCE FACULTY POSITIONS

Australian National University | Research School of Computer Science

The ANU College of Engineering and Computer Science (CECS) is undergoing significant change and expansion through the Reimagine investment. This substantial 15-year, commitment will transform traditional engineering and computer science disciplines for the 21st century.

This is an exciting time to join our Faculty and be part of a community that prides itself on delivering cutting-edge research and research-led education to develop future leaders, who will find solutions to some of the world's greatest technological and social challenges.

To enquire about these positions please contact Computer Science Director, Professor Tony Hosking, via email to director.rscs@anu.edu.au

Come and enjoy the fantastic Australian lifestyle, while working for a world leading University with outstanding staff benefits, including;

- 17% superannuation
- 26 weeks paid parental leave
- 4 weeks paid annual leave
- Exceptional Professional Development opportunities

www.anu.edu.au/jobs



The application material should include a full academic CV, a personal narrative on teaching and research, a list of at least three references, and up to three sample publications. Unique to applications this year to CSE, the personal narrative should include one highlighted paragraph that describes the applicant's most significant achievement and includes a reference to a single publication or preprint that the search committee will review in detail. For full consideration, applications are due by December 15, 2019.

Georgia Tech is an Affirmative Action/Equal Opportunity Employer. Applications from women and under-represented minorities are strongly encouraged. For more information about Georgia Tech's School of Computational Science and Engineering please visit: <http://www.cse.gatech.edu/>.

The Johns Hopkins University **Department of Computer Science** **Tenure-Track Faculty Positions**

The Johns Hopkins University's Department of Computer Science seeks applicants for tenure-track faculty positions at all levels and across all areas of computer science. The department will consider offers in two tracks: (1) an open track seeking excellent candidates across all areas of computer science; and (2) a track seeking candidates in the areas of human computer interaction (HCI), human AI interaction, computational health, artificial intelligence and machine learning. The search will focus on candidates at the junior level, but all qualified applicants will be considered.

Plans for faculty growth in the department are aligned with School and University initiatives in health and AI. Additionally, the faculty will continue to grow by adding excellent and diverse candidates across all areas of computer science. The HCI search is part of a newly-launched initiative (<http://hci.jhu.edu>) that seeks to transform existing HCI research activities across the university by making several faculty hires within Computer Science.

The Department of Computer Science has 31 full-time tenured and tenure-track faculty members, 8 research and 5 teaching faculty members, 200 PhD students, 200 MSE/MSSI students, and over 500 undergraduate students. There are several affiliated research centers and institutes including the Laboratory for Computational Sensing and Robotics (LCSR), the Center for Language and Speech Processing (CLSP), the JHU Information Security Institute (JHISI), the Institute for Data Intensive Engineering and Science (IDIES), the Malone Center for Engineering in Healthcare (MCEH), the Institute for Assured Autonomy (IAA), and other labs and research groups. More information about the Department of Computer Science can be found at www.cs.jhu.edu and about the Whiting School of Engineering at <https://engineering.jhu.edu>.

Applicants should submit a curriculum vitae, a research statement, a teaching statement, three recent publications, and complete contact information for at least three references.

Applications must be made on-line at <http://apply.interfolio.com/69225>. While candidates who complete their applications by December 15, 2019 will receive full consideration, the department will consider applications submitted after that date.

The Whiting School of Engineering and the Department of Computer Science are committed to building a diverse educational environment: <https://www.cs.jhu.edu/diversity/>.

The Johns Hopkins University is committed to equal opportunity for its faculty, staff, and students. To that end, the University does not discriminate on the basis of sex, gender, marital status, pregnancy, race, color, ethnicity, national origin, age, disability, religion, sexual orientation, gender identity or expression, veteran status or other legally protected characteristic. The University is committed to providing qualified individuals access to all academic and employment programs, benefits and activities on the basis of demonstrated ability, performance and merit without regard to personal factors that are irrelevant to the program involved.

Southern University of Science and Technology (SUSTech) **Professor Position in Computer Science and Engineering**

The Department of Computer Science and Engineering (CSE, <http://cse.sustc.edu.cn/en/>), Southern University of Science and Technology (SUSTech) has multiple Tenure-track faculty openings at all ranks, including Professor/Associate Professor/Assistant Professor. We are looking for outstanding candidates with demonstrated research achievements and keen interest in teaching, in the following areas (but are not restricted to):

- Data Science
- Artificial Intelligence
- Computer Systems (including Networks, Cloud Computing, IoT, Software Engineering, etc.)
- Cognitive Robotics and Autonomous Systems
- Cybersecurity (including Cryptography)

Applicants should have an earned Ph.D. degree and demonstrated achievements in both research and teaching. The teaching language at SUSTech is bilingual, either English or Putonghua. It is perfectly acceptable to use English in all lectures, assignments, exams. In fact, our existing faculty members include several non-Chinese speaking professors.

The Department of Computer Science and Engineering at SUSTech was founded in 2016. It has 33 professors, all of whom hold doctoral degrees or have years of experience in overseas universities. Among them, three are IEEE fellows; one IET fellow. The department is expected to grow to 50 tenure track faculty members eventually, in addition to teaching-only professors and research-only professors.

Established in 2012, the Southern University of Science and Technology (SUSTech) is a public institution funded by the municipal of Shenzhen, a special economic zone city in China. Shenzhen is a major city located in Southern China, situated immediately north to Hong Kong Special Administrative Region. As one of China's major gateways to the world, Shenzhen is the country's fastest-growing city in the past two decades. The city is the high-tech and manufacturing hub of southern China, home to the world's third-busiest container port, and the fourth-busiest airport on the Chinese mainland. Shenzhen ranks 66th place on the 2017 Global City Competitiveness List, released by the National Academy of Economic Strategy, the Chinese Academy of Social Sciences and United Nations Habitat. By the end of 2018, there were around 20 million residents in Shenzhen.

SUSTech is committed to increase the diversity of its faculty and has a range of family-friendly policies in place. The university offers competitive salaries and fringe benefits including medical insurance, retirement and housing subsidy, which are among the best in China.

To apply, please provide a cover letter identifying the primary area of research, curriculum vitae, and research and teaching statements, and forward them to cshire@sustc.edu.cn.

University of Illinois at Chicago **Lecturer Non-Tenure Track / Computer Science**

The Computer Science Department at the University of Illinois at Chicago is seeking full-time teaching faculty members. The Lecturer teaching track is a long-term career track that starts with the Lecturer position and offers opportunities for advancement to Senior Lecturer. Candidates would be working alongside 15 full-time teaching faculty with over 150 years of combined teaching experience and 12 awards for excellence. The department seeks candidates dedicated to teaching; candidates must have evidence of effective teaching; or present a convincing case of future dedication and success in the art of teaching. Content areas of interest include introductory programming, data structures, computer organization/systems, web development, data science, software engineering, and machine learning. The standard teaching load is 1-3 undergraduate courses per semester (depending on course enrollment).

The University of Illinois at Chicago (UIC) is one of the top-ten most diverse universities in the US (US News and World Report), a top-ten best value (Wall Street Journal and Times Higher Education) and a Hispanic serving institution. UIC's hometown of Chicago epitomizes the modern, livable, vibrant city. Located on the shore of Lake Michigan, Chicago offers an outstanding array of cultural and culinary experiences. As the birthplace of the modern skyscraper, Chicago boasts one of the world's tallest and densest skylines, combined with an 8100-acre park system and extensive public transit and biking networks.

Minimum qualifications include an MS in Computer Science or a closely related field or appropriate graduate degrees for specific course material (e.g., computer ethics), and either (a) demonstrated evidence of effective teaching, or (b) convincing argument of future dedication and success in the art of teaching. Applications are submitted online at <https://jobs.uic.edu/>. In the online application include a curriculum vitae, names and addresses of at least three references, a statement providing evidence of effective teaching, a statement describing your past experience in activities that promote diversity and inclusion (or plans to make future contributions), and recent teaching evaluations. For additional information contact Professor Mitch Theys, Committee Chair, mtheys@uic.edu.

For fullest consideration, please apply by October 15, 2019. We will continue to accept and review applications until the positions are filled.

The University of Illinois is an Equal Opportunity, Affirmative Action employer. Minorities, women, veterans and individuals with disabilities are encouraged to apply. The University of Illinois conducts background checks on all job candidates upon acceptance of contingent offer of employment. Background checks will be performed in compliance with the Fair Credit Reporting Act.

University of Kentucky

Department of Computer Science

Tenure-Track Faculty Position

The Department of Computer Science at the University of Kentucky invites applications for a tenure-track faculty position to begin August 2020. We seek excellent candidates in all areas, with a particular desire for expertise in data science and the related areas of machine learning, big data, cloud computing, and data mining. We will consider all ranks, with preference for candidates at the assistant professor level. Outstanding candidates at the rank of assistant professor will be considered for an endowed fellowship.

We are seeking energetic and creative faculty who have a passion for teaching students. Candidates should be able to teach courses in our newly established graduate program in Data Science, upper-level courses in their areas of expertise, and, ultimately, core courses in our ABET-accredited undergraduate Computer Science program.

We favor researchers who are eager to collaborate to solve problems that extend beyond their own research areas. Our faculty members have established research programs with other members of the department and with a wide variety of other departments and programs, including statistics, biology, linguistics, internal medicine, electrical engineering, computer engineering, and the humanities.

The Department, one of the first computer science departments in the United States, has 22 faculty members committed to excellence in education, research, and service. We offer programs leading to the Bachelors, Masters, and Ph.D. degrees. The University of Kentucky is located in Lexington, the scenic heart of the Bluegrass Region of Kentucky. With recognition as one of the safest, most creative, and well-educated cities in the nation, Lexington is an ideal location to build an outstanding, work-life balanced career.

Candidates must have earned a PhD in Computer Science or closely related field at the time employment begins. To apply, a University of Kentucky Academic Profile must be submitted at the following link: <http://ukjobs.uky.edu/postings/250618>.

Applications are now being accepted. Review of credentials will begin immediately and continue until the positions are filled.

Questions should be directed to HR/Employment by phone at 1-859-257-9555 press 2 or email (ukjobs@email.uky.edu), or to Diane Mier (diane.mier@uky.edu) in the Computer Science Department.

Upon offer of employment, successful applicants must undergo a national background check as required by University of Kentucky Human Resources. The University of Kentucky is an equal opportunity employer and especially encourages applications from minorities and women.

University of Maryland

Tenure-Track Faculty in Electrical & Computer Engineering

The Department of Electrical and Computer Engineering at the University of Maryland, College Park, invites applications for multiple tenure-track faculty positions. Applicants should have a demonstrated record of research achievements

and publications in one of the following areas:

a) **Machine learning algorithms and applications** with methodologies from signal processing, computer vision, and/or optimization;

b) **Internet of Things (IoT)** broadly defined, including hardware/software methodologies of scalable, secure and resilient infrastructure for connected smart devices, data analytics, edge and cloud computing; and

c) **Quantum information processing and technology**, particularly with a focus on applications at the intersection of engineering and physical science.

The tenure-track appointments are expected to be at the Assistant Professor level, although more senior candidates with outstanding records of research achievements may be considered. Successful applicants will be expected to maintain active research programs and teach undergraduate and graduate courses in Electrical and Computer Engineering.

A Ph.D. in Electrical/Computer Engineering or a related discipline is required. Candidates should be creative and adaptable and should have a high potential for both research and teaching.

For best consideration, applications should be submitted by **10 December 2019** online at <https://ejobs.umd.edu> (position number 125223). An application should include a cover letter, curriculum vitae, a list of at least three references, examples of research achievements including three significant publications, a research statement (not exceeding three pages), and a statement of teaching philosophy (not exceeding two pages). The applicant should clearly indicate in the cover letter which of the above areas best fits the candidate's research interests.

University of Illinois at Urbana-Champaign

Full-Time Faculty Positions

The Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign (ECE ILLINOIS) invites applications for full-time faculty positions. Senior and mid-career faculty are encouraged to apply but all qualified candidates will be considered at all levels and in all areas of electrical and computer engineering, broadly defined, with particular emphasis in the areas of Computational and Physical Aspects of Electronics and Electromagnetics; Power and Energy Systems; Quantum Information Science; Bioacoustics and Bioimaging; Circuits - System on a Chip; AI/Autonomous Systems; Robotics; Machine Vision; Embedded Computing Systems and the Internet of Things; Data-Centric Computer Systems and Storage; Networked and Distributed Computing Systems. Applications are encouraged from candidates whose degrees and research programs are in core as well as broad interdisciplinary areas of electrical and computer engineering. Ideal candidates include those who demonstrate evidence of a commitment to diversity, equity, and inclusion through research, teaching, and/or service endeavors.

Of particular interest is quantum information science, including quantum computing, quantum communication, quantum sensing, and the development of novel hardware and architectures for quantum information processing.

This position is part of a \$15M investment in the new Illinois Quantum Information Science and Technology (QUIST) center, which also includes multiple hires in Physics, Computer Science, and Mathematics.

Applicants for all positions at the assistant professor level must have an earned Ph.D. or equivalent degree, excellent academic credentials, and an outstanding ability to teach effectively at both the graduate and undergraduate levels. Successful candidates will be expected to initiate and carry out independent research and to perform academic duties associated with our B.S., M.S., and Ph.D. programs.

Qualified senior candidates may also be considered for tenured Associate and Professor positions as part of the Grainger Engineering Breakthroughs Initiative (GEBI), which is backed by a \$100-million gift from the Grainger Foundation.

ECE ILLINOIS is in a period of intense demand and growth, serving over 3000 students and averaging 7 new tenure-track faculty hires per year in recent years. Faculty in the department carry out research in a broad spectrum of areas and are supported by world-class interdisciplinary research facilities, including the Coordinated Science Laboratory, the Information Trust Institute, the Parallel Computing Institute, the Nick Holonyak Jr. Micro and Nanotechnology Laboratory, the Beckman Institute for Advanced Science and Technology, the Carl R. Woese Institute for Genomic Biology, as well as several industrial centers and programs that foster international collaborations. The ECE Department also supports and encourages faculty involvement with the Nation's first engineering-based College of Medicine that has opened on campus. The plans are to facilitate transition from engineering breakthroughs into translational medical practices. The department has one of the very top programs in the world, granting approximately 450 B.S. degrees, 100 M.S. degrees, 80 M.Eng. degrees, and 75 Ph.D. degrees annually. ECE ILLINOIS is housed in its new 235,000 sq. ft. building designed to set a new standard in energy efficiency which is a major campus addition with minimal carbon footprint.

In order to ensure full consideration by the Search Committee, applications must be received by December 1, 2019. Applications will be reviewed until suitable candidates are identified. Interviews and offers may take place before the deadline but all applications received by the deadline would receive full consideration. Salary will be commensurate with qualifications. Preferred starting date is August 16, 2020, but is negotiable. Applications can be submitted by going to <http://jobs.illinois.edu> and uploading a cover letter, CV, research statement, teaching statement, and statement on commitment to diversity, along with names of three references. The statement on diversity should address past and/or potential contributions to diversity, equity, and inclusion through research, teaching, and/or service. For inquiry, please call 217-333-2302 or email ece-recruiting@illinois.edu.

The University of Illinois conducts criminal background checks on all job candidates upon acceptance of a contingent offer.

The University of Illinois is an Equal Opportunity, Affirmative Action employer. Minorities, women, veterans and individuals with disabilities

are encouraged to apply. For more information, visit <http://go.illinois.edu/EEO>. To learn more about the University's commitment to diversity, please visit <https://engineering.illinois.edu/about/diversity.html>.

We have an active and successful dual-career partner placement program and a strong commitment to work-life balance and family-friendly programs for faculty and staff (<http://provost.illinois.edu/faculty-affairs/work-life-balance/>).

University of Memphis

Department of Computer Science

Assistant Professor

The Department of Computer Science at the University of Memphis is seeking candidates for Assistant Professor position(s) beginning Fall 2020. Exceptionally qualified candidates in all areas of computer science are invited, while candidates with core expertise in emerging areas of edge computing, machine learning, and software engineering are particularly encouraged to apply. Successful candidates are expected to develop externally sponsored research programs, lead or participate in collaborative research projects within Computer Science and beyond, teach both undergraduate and graduate courses and provide academic advising to students at all levels. Candidates from minority and underrepresented groups are highly encouraged to apply.

Applicants should hold a PhD in Computer Science, or related discipline, and be committed to excellence in both research and teaching. Salary is highly competitive and dependent upon qualifications.

The Department of Computer Science (www.memphis.edu/cs) offers B.S., M.S., and Ph.D. programs as well as graduate certificates in Data Science and Information Assurance. The Department has been ranked 55th among CS departments with federally funded research. The Department regularly engages in large-scale multi-university collaborations across the nation. For example, CS faculty lead the NIH-funded Big Data "Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K)" and the "Center for Information Assurance" (CfIA). In addition, CS faculty work closely with multidisciplinary centers at the university such as the "Institute for Intelligent Systems" (IIS).

Well known as America's distribution hub, Memphis was ranked as America's 6th best city for jobs by Glassdoor in 2017. Memphis metropolitan area has a population of 1.3 million. It boasts a vibrant culture and has a pleasant climate with an average temperature of 63 degrees.

Screening of applications begins immediately. For full consideration, application materials should be received by December 1, 2019. However, applications will be accepted until the search is completed.

To apply, please visit <https://workforum.memphis.edu/postings/23648>. Include a cover letter, curriculum vitae, teaching and research statements, and three letters of recommendation. Direct all inquiries to Corinne O'Connor (cconnor2@memphis.edu).

A background check will be required for employment. The University of Memphis is an Equal Opportunity/Equal Access/Affirmative

Action employer committed to achieving a diverse workforce.

The University of Nevada, Las Vegas

Assistant/Associate Professor in Systems and Assistant/Associate Professor in Big Data and Machine Learning

The University of Nevada, Las Vegas invites applications for Assistant/Associate Professor in Systems [R0118463] and Assistant/Associate Professor in Big Data and Machine Learning [R0118464], Department of Computer Science UNLV Howard R. Hughes College of Engineering

ROLE OF THE POSITIONS

The Department of Computer Science at the University of Nevada, Las Vegas (UNLV) invites applications for full-time, tenure-track Assistant Professors, or tenure-track/tenured Associate Professors in Systems and Big Data and Machine Learning commencing Fall 2020. A distinguished record in undergraduate and graduate computer science education and well-funded scholarly research will be required for appointment as an Associate Professor with tenure.

QUALIFICATIONS

These positions require a Ph.D. in Computer Science from a regionally accredited college or university at the time of appointment. The successful candidate should have a good understanding of theory as well as practice.

Systems: He/she must be able to teach a practical course in Operating Systems (3rd year) as well as Programming Languages (also 3rd year). In addition, preference will be given to candidates with experience in one or more of the following areas: Compiler and language construction/implementation, Formal verification (e.g., CSP), Distributed systems and parallel programming.

Big Data and Machine Learning: He/she must be able to teach courses in Big Data as well as Machine Learning. In addition, preference will be given to candidates with experience in one or more of the following areas: Distributed and Parallel Learning, Reinforcement Learning, MapReduce, and Deep Learning.

APPLICATION DETAILS

Submit a letter of interest, a detailed resume listing qualifications and experience, and the names, addresses, and telephone numbers of at least three professional references who may be contacted. Applicants should fully describe their qualifications and experience, with specific reference to each of the minimum and preferred qualifications because this is the information on which the initial review of materials will be based.

Although these positions will remain open until filled, review of candidates' materials will begin on January 16, 2020 and best consideration will be gained for materials submitted prior to that date. Materials should be addressed to Dr. Matt Pedersen, Search Committee Chair, and are to be submitted online <https://www.unlv.edu/jobs> as we do not accept emailed materials. For assistance with the application process, please contact UNLV Human Resources at (702) 895-3504 or applicant.inquiry@unlv.edu.

EEO/AA/Vet/Disability Employer

Vanderbilt University

Tenure Track Faculty Position in Computer Science And Data Science

THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE (EECS) AT VANDERBILT UNIVERSITY is seeking candidates for a tenure-track faculty position in computer science. Candidates will be considered in broadly defined areas of machine learning, artificial intelligence, data science, natural language processing, visualization, and/or computer vision. Successful candidates are expected to teach at the undergraduate and graduate levels and to develop and grow vigorous programs of externally funded research. The Vanderbilt CS program provides a unique, collaborative, and interdisciplinary research environment. These positions are part of a broader multi-year faculty hiring initiative in the data sciences. Candidates will be core members of Vanderbilt's recently established Data Science Institute, and will be expected to contribute to the research and teaching mission of the institute, and ideally to foster research collaborations with existing faculty across academic departments working on data science related topics.

Vanderbilt University is a private, internationally renowned research university located in vibrant Nashville, Tennessee. Its 10 schools share a single cohesive campus that nurtures interdisciplinary activities. The School of Engineering is on a strong upward trajectory in national and international stature and prominence, and has built infrastructure to support a significant expansion in faculty size. In the rankings of graduate engineering programs by U.S. News & World Report, the School ranks in the top 5 among programs with fewer than 100 faculty members. 5-year average T/TK faculty funding in the EECS Department is above \$800k per year. All junior faculty members hired during the past 15 years have received prestigious young investigator awards, such as NSF CAREER and DARPA CSSG.

With a metro population of approximately 1.5 million people, Nashville has been named one of the 15 best U.S. cities for work and family by Fortune magazine, was ranked as the #1 most popular U.S. city for corporate relocations by Expansion Management magazine, and was named by Forbes magazine as one of the 25 cities most likely to have the country's highest job growth over the coming five years. Major industries include tourism, printing and publishing, manufacturing technology, music production, higher education, finance, insurance, automobile production and health care management.

Vanderbilt University has a strong institutional commitment to recruiting and retaining an academically and culturally diverse community of faculty. Minorities, women, individuals with disabilities, and members of other underrepresented groups, in particular, are encouraged to apply. Vanderbilt is an Equal Opportunity/Affirmative Action employer.

Applications should be submitted on-line at: <http://apply.interfolio.com/69338>. For more information, please visit our web site: <http://engineering.vanderbilt.edu/eeecs>. Applications will be reviewed on a rolling basis beginning November 15, 2019 with telephone interviews beginning December 1, 2019. The final application deadline is January 15, 2020.

Providing Sound Foundations for Cryptography

On the work of Shafi Goldwasser and Silvio Micali

Cryptography is concerned with the construction of schemes that withstand any abuse. A cryptographic scheme is constructed so as to maintain a desired functionality, even under malicious attempts aimed at making it deviate from its prescribed behavior. The design of cryptographic systems must be based on firm foundations, whereas ad hoc approaches and heuristics are a very dangerous way to go. These foundations were developed mostly in the 1980s, in works that are all co-authored by Shafi Goldwasser and/or Silvio Micali. These works have transformed cryptography from an engineering discipline, lacking sound theoretical foundations, into a scientific field possessing a well-founded theory, which influences practice as well as contributes to other areas of theoretical computer science.

This book celebrates these works, which were the basis for bestowing the 2012 A.M. Turing Award upon Shafi Goldwasser and Silvio Micali. A significant portion of this book reproduces some of these works, and another portion consists of scientific perspectives by some of their former students. The highlight of the book is provided by a few chapters that allow the readers to meet Shafi and Silvio in person. These include interviews with them, their biographies and their Turing Award lectures.

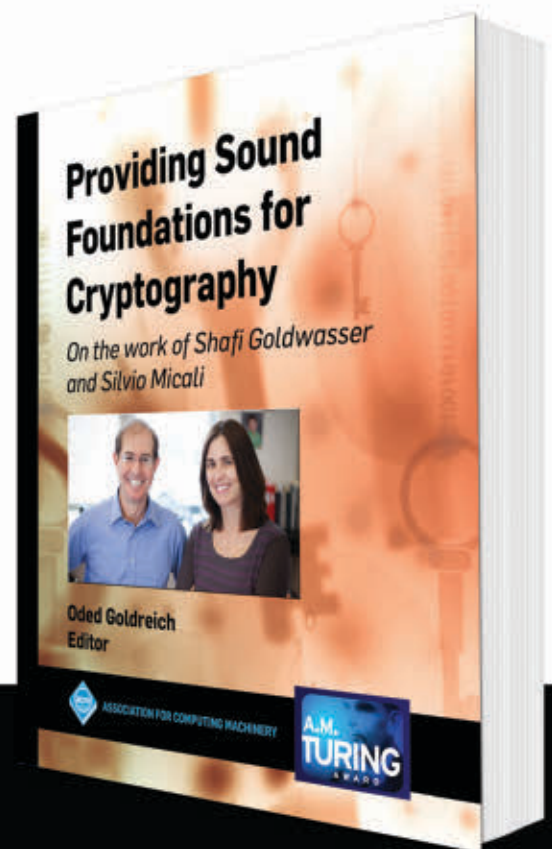
Oded Goldreich, Editor

ISBN: 978-1-4503-7267-1

DOI: 10.1145/3335741

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



ACM BOOKS
Collection II

[CONTINUED FROM P. 96] **throughout your career. Can you tell me about the Parallel Data Lab, which you founded in 1993, two years after you left Berkeley and moved to Carnegie Mellon University?**

At Berkeley, I loved the benefits of interacting with industry—engaging with smart people and real-world problems and trends. I wanted to build the same infrastructure at Carnegie Mellon. We created what we called the Parallel Data Consortium, which was a vehicle to give companies access and interaction with the people in the lab. Initially, we held annual retreats in which the research results of the lab were shared and discussed with industry collaborators. That structure has evolved over time, as have the industry participants. But the number of companies involved in it grew from an initial six to about 20 now, plus strong engagements with government funding agencies.

One of the more impactful projects to emerge from the lab was Network Attached Secure Disks (NASD) technology, which moved magnetic storage disks out of the host computers and communicated with them via networking protocols in the interests of providing more scalable storage.

We did a few things that ended up having influence on the architecture. The first was to call up the interface abstraction from the magnetic disk layer upward in the stack, but not all the way to the file system abstraction. That's still the way it's done today. Almost all of the large file systems have an interface layer, whether it's an AWS object or a file system object. It's usually separate from the file system above it, and it is the scalability component.

The security aspect is also interesting, because we separated policy from implementation. The storage object would implement security and access control using a Merkle tree protection, a lot like Blockchain is used in the integrity of ledgers. That gave us the ability to move storage across the network in a way where the storage components didn't have to understand the file systems' policies. It was a big influence on the distributed systems community.

In 2018, you joined another consortium of industry and academia, Toronto's Vector Institute, as president and CEO.

If I compare it to the Parallel Data Lab, what I say to people at CMU is, it's kind of like the Parallel Data Lab, except it's moved outside the university, and it's dealing with 10 times as much people, and 10 times the amount of money. And a lot more partners.

How have machine learning, deep learning, and the other areas the Vector Institute is looking at impacted systems design?

Around 2010, Carlos Guestrin at Carnegie Mellon and Joseph Hellerstein at Berkeley began to think about solving for machine learning problems. And that meant distributive processing. What happens when we try to do an iterative convergent solution of an equation using distributed systems? And how are we going to deal with all of that communication? And that became the big way that machine learning impacted systems design. We are going to do so much communication that the algorithms are going to need to explicitly take into consideration the cost of communication.

You've also done some work in that area with Eric Xing and Greg Ganger.

In 2012, we started working on stale synchronous processing, or SSP, as opposed to bulk synchronous processing, which is the basis for parallel computing that Leslie Valiant did in his Turing Award work. The idea in stale synchronous is that when you are searching for an approximate solution—which is what a convergent algorithm is, because you're going to assume that it's close enough—you can allow error as long as you can bound it. In particular, you can allow some signals to arrive later than others. In the traditional computing world, we would have called this relaxed consistency.

My students since then have said that, while it's true that staleness can be tolerated, it isn't necessarily fast. So, the work has been more along the lines of, how can we allow staleness if it increases the speed of the system, while maximizing freshness in order that we converge fast? 

Leah Hoffmann is a technology writer based in Piermont, NY, USA.

© 2019 ACM 0001-0782/19/12 \$15.00



Association for
Computing Machinery

Digital Government: Research and Practice

Digital Government: Research and Practice (DGOV) is an interdisciplinary journal on the potential and impact of technology on governance innovations and its transformation of public institutions. It promotes applied and empirical research from academics, practitioners, designers, and technologists, using political, policy, social, computer, and data sciences methodologies.



For further information
or to submit your
manuscript,
visit dgov.acm.org

Q&A

RISCy Beginnings

In a career launched by groundbreaking research, Garth Gibson continues to shepherd technological advances “from blackboard through standards and to commercial reality.”

GARTH GIBSON HAS spent his career pushing data storage systems to higher levels of performance, reliability, and scalability. While he was a graduate student at the University of California, Berkeley, Gibson was a part of groundbreaking research on Redundant Arrays of Inexpensive Disks (RAID). Later, as a professor at Carnegie Mellon University, he worked on projects like Network Attached Secure Disk (NASD) technology, and a clustered storage system used by Roadrunner, the world's first peta-scale supercomputer. Here, he speaks about handling failures, collaborating with industry and academia, and how deep learning has impacted systems design.

When you were a graduate student at the University of California at Berkeley, you joined a computer architecture team led by David Patterson that was building a complete system based on RISC concepts. How did you get started on storage?

David Patterson walked into my office around 1986 and said, “You know, I think we’re doing really well with the development of the computer architecture and data processing. Let’s think about the broader systems issues.” There were two areas of interest. One was in networking distributed systems, and the other was in storage. So I started trying to figure out what was going on in storage, and what the most important issues were when it came to storing and accessing huge amounts of information.



We also knew we wanted to build a parallel storage system, just like we wanted to build multiprocessors into highly parallel computers.

One of the dominant problems was that, if your storage medium—the magnetic disk—failed, you put everything on the computer at risk.

The database community, by that point, already had their answer, namely duplicates. Memory systems researchers had also developed error-cracking code that stretched across memory chips to cope with the loss of the memory system. Then we noticed a group that pointed out that there’s a big difference between a code that can detect a loss and replace it, and a code that doesn’t have to detect, but just replaces a known loss. And we realized that the most important failures were going to be identifiable, so we built the taxonomy around erasure codes. The original paper on re-

dundant arrays of inexpensive disks, or RAID, presented the taxonomy, the problem space, the goals in parallelism, and a way to compare the different ways that you could do it.

Your co-authors, David Patterson and Randy Katz, then built a research program to develop those ideas, which were eventually commercialized under a slightly different acronym: “redundant arrays of independent disks.”

We made a very important strategic decision to engage the industry, which was Patterson’s approach to doing research projects. Because we chose not to patent or trademark the ideas, they were quickly adopted by industry. We called it “inexpensive” because replacing one unit with 100 units was never going to be a good idea if it meant your product was 100 times the cost. But nobody likes to try to sell something by saying, “This is a high-value, high-cost product, and that’s why it’s called ‘inexpensive.’” Fortunately, the word “independent” is entirely appropriate, because the only reason that RAID’s failure modeling works is that the failure domains of each drive are independent failures.

In any event, when I look at data-sheets for a computer system today, I see that the chips that are built into the motherboards have support for RAID 0, RAID 1, RAID 5, and RAID 10.

Like Patterson, you have gone back and forth between industry and academia [CONTINUED ON P. 95]



IDC 2020

DESIGNING
FOR THE FUTURE

ACM Interaction Design & Children Conference June 21 - 24, 2020 in London (UK)

IDC 2020 will be the 19th in an annual series that encourages research concentrating on the design, development, and use of interactive technologies for children. Ambitious and idea-driven researchers, designers, and educators from all over the world will gather for IDC to share and discuss these ideas. This year's theme is 'Designing for the future'.

IDC 2020 welcomes the following submissions:

- | | |
|----------------------------------|-------------------|
| ✓ R&D Competition (Child Ideas) | <i>9 December</i> |
| ✓ Full and Short Papers | <i>20 January</i> |
| ✓ Workshop Organizer Submissions | <i>3 February</i> |
| ✓ Course Organizer Submissions | <i>2 March</i> |
| ✓ R&D Competition (Adult Ideas) | <i>9 March</i> |
| ✓ Doctoral Consortium | <i>30 March</i> |
| ✓ Works-in-Progress, Demo & Art | <i>6 April</i> |

We look forward to seeing all of you here in London!



Association for
Computing Machinery

IDC.ACM.ORG/2020

