

COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

12/2017 VOL.60 NO.12

Cyber Security

Nuclear Security

Illogical Logic

Computing Is the Secret Ingredient
The Death of Big Software
Bitcoin's Academic Pedigree
Q&A with John Hennessy



32nd IEEE International Parallel and Distributed Processing Symposium

May 21-25, 2018
Vancouver, British Columbia
CANADA

ANNOUNCING 23 PLANNED WORKSHOPS

IPDPS Workshops Monday 21 May 2018

| | |
|-------------|--|
| HCW | Heterogeneity in Computing Workshop |
| RAW | Reconfigurable Architectures Workshop |
| HiCOMB | High Performance Computational Biology |
| GABB | Graph Algorithms Building Blocks |
| EduPar | NSF/TCPP W. on Parallel and Distributed Computing Education |
| HIPS | High Level Programming Models and Supportive Environments |
| HPBDC | High-Performance Big Data, Deep Learning, and Cloud Computing |
| AsHES | Accelerators and Hybrid Exascale Systems |
| PDCO | Parallel / Distributed Computing and Optimization |
| HPPAC | High-Performance, Power-Aware Computing |
| APDCM | Advances in Parallel and Distributed Computational Models |
| ParLearning | Parallel and Distributed Computing for Large-Scale Machine Learning and Big Data Analytics |

IPDPS Workshops Friday 25 May 2018

| | |
|-----------|---|
| CHIWW | Chapel Implementers and Users Workshop |
| PDSEC | Parallel and Distributed Scientific and Engineering Computing |
| JSSPP | Job Scheduling Strategies for Parallel Processing |
| iWAPT | International Workshop on Automatic Performance Tunings |
| ParSocial | Parallel and Distributed Processing for Computational Social Systems |
| GraML | Graph Algorithms and Machine Learning |
| CEBDA | Convergence of Extreme Scale Computing and Big Data Analysis |
| MPP | Parallel Programming Model: Special Edition on Edge/Fog/In-Situ Computing |
| PASCO | Parallel Symbolic Computation |
| PMAW | Programming Models and Algorithms Workshop |
| ROME | Runtime and Operating Systems for the Many-core Era |

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**. When a workshop announces its Call for Papers, the link on the IPDPS Workshops webpage is activated, and the call for papers submission due date is posted. Proceedings of the workshops are distributed at the conference and are submitted for inclusion in the IEEE Xplore Digital Library after the conference.

GENERAL CHAIR

Bora Uçar (CNRS and ENS Lyon, France)

PROGRAM CHAIR and VICE-CHAIR

Anne Benoît (ENS Lyon, France) and
Ümit V. Çatalyürek (Georgia Institute of Technology, USA)

WORKSHOPS CHAIR and VICE-CHAIR

Erik Saule (University of North Carolina at Charlotte, USA) and
Jaroslav Zola (University at Buffalo, USA)

STUDENT PARTICIPATION CHAIR and VICE-CHAIR

Trilce Estrada (University of New Mexico, USA) and
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

IPDPS extends a special invitation for companies to become an IPDPS 2018 Industry Partner and to share in the benefits of associating with an international community of top researchers and practitioners in fields related to parallel processing and distributed computing. Visit the IPDPS website to see ways to participate.

IPDPS 2018 VENUE

Rising against a backdrop of majestic coastal mountains on the Pacific Northwest coast, the JW Marriott Parq Vancouver is located in the heart of downtown Vancouver's urban entertainment and resort complex. IPDPS 2018 attendees will enjoy state of the art meeting facilities, with Vancouver as a jumping off point for some of the world's grand sightseeing adventures.

For details, visit www.ipdps.org

IMPORTANT DATES

- Conference Preliminary Author Notification **December 8, 2017**
- Workshops' Call for Papers Deadlines **Most Fall After December 8, 2017**

IEEE computer society

Sponsored by IEEE Computer Society
Technical Committee on Parallel Processing



In cooperation with
ACM SIGARCH & SIGHPC and IEEE TCCA & TCDDP

2018 Artificial Intelligence · Blockchain · Cloud

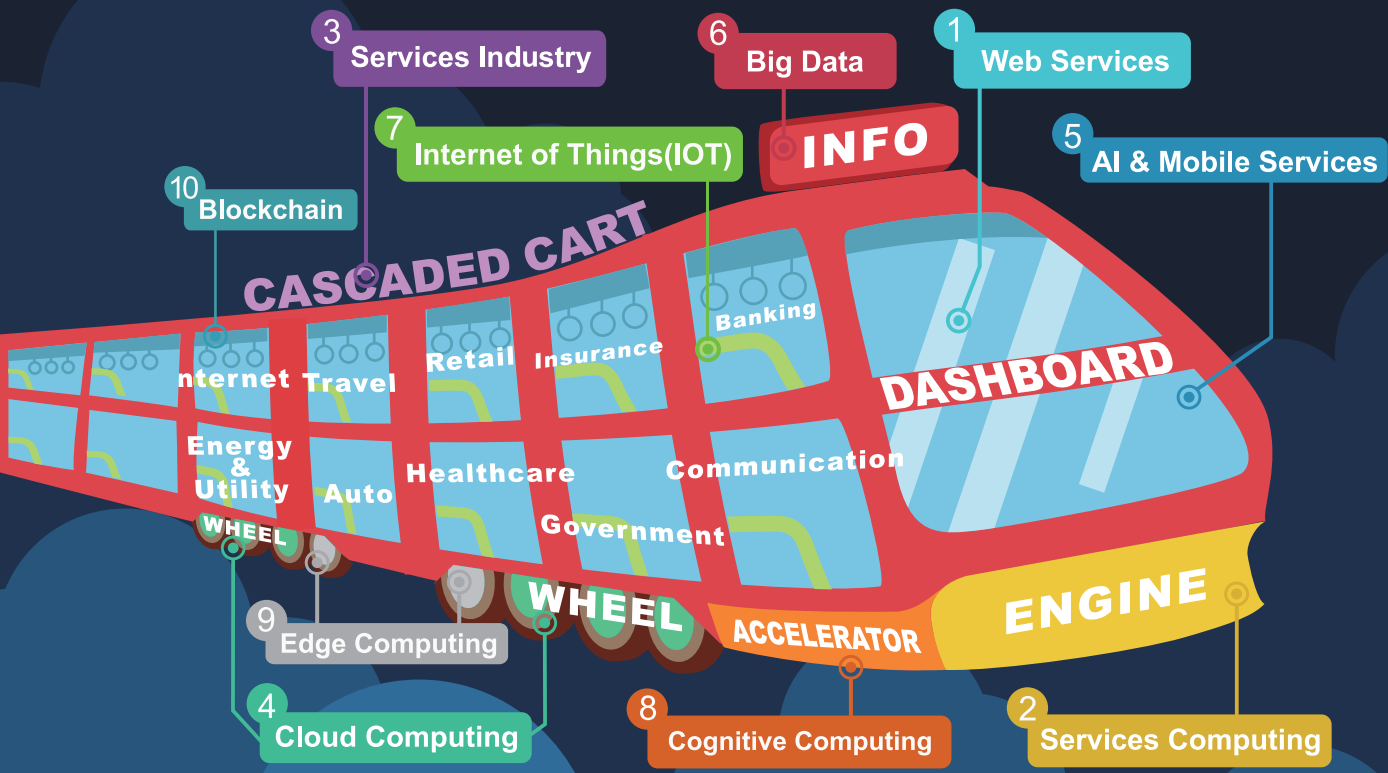
CREATING THE FUTURE

June 25 - June 30, Seattle, USA

CELEBRATING THE 25th GATHERING OF ICWS



- 1 The 25th International Conference on Web Services (**ICWS 2018**)
- 2 The 15th International Conference on Services Computing (**SCC 2018**)
- 3 The 14th World Congress on Services (**SERVICES 2018**)
- 4 The 11th International Conference on Cloud Computing (**CLOUD 2018**)
- 5 The 7th International Conference on AI & Mobile Services (**AIMS 2018**)
- 6 The 7th International Congress on Big Data (**BigData Congress 2018**)
- 7 The 3rd International Congress on Internet of Things (**ICIOT 2018**)
- 8 The 2nd International Conference on Cognitive Computing (**ICCC 2018**)
- 9 The 2nd International Conference on Edge Computing (**EDGE 2018**)
- 10 The 1st International Conference on Blockchain (**ICBC 2018**)



Submission Deadlines

- | | |
|--|--|
| 2/6/2018: ICWS 2018 (http://icws.org) | 2/28/2018: BigData Congress 2018 (http://BigDataCongress.org) |
| 2/21/2018: SCC 2018 (http://theSCC.org) | 3/17/2018: ICIOT 2018 (http://iciot.org) |
| 2/25/2018: SERVICES 2018 (http://ServicesCongress.org) | 3/17/2018: ICC 2018 (http://theCognitiveComputing.org) |
| 2/6/2018: CLOUD 2018 (http://theCloudComputing.org) | 3/17/2018: EDGE 2018 (http://theEdgeComputing.org) |
| 2/21/2018: AIMS 2018 (http://ai1000.org) | 3/17/2018: ICBC 2018 (http://Blockchain1000.org) |

Contact: ZHANGLJ@ACM.ORG
 Dr. Liang-Jie (LJ) Zhang
 Steering Committee Chair

Largest not-for-profits organization (501(c)(3))
 dedicated for serving 30,000+ worldwide
 services computing professionals



Departments

- 5 **Editor's Letter**
Computing Is the Secret Ingredient (well, not so secret)
By Andrew A. Chien
-
- 6 **Letters to the Editor**
Start CS Students Off with Industry Best Practices
-
- 7 **Cerf's Up**
Now for Something Entirely Different
By Vinton G. Cerf
-
- 8 **BLOG@CACM**
Building Tools to Help Students Learn to Program
Philip Guo summarizes his first three years of research into building tools to support those learning computer programming.
-
- 23 **Calendar**
-
- 98 **Careers**

Last Byte

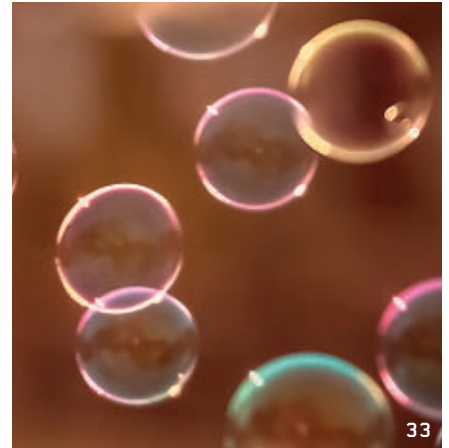
- 112 **Q&A**
Grooming the Leaders of Tomorrow
By Leah Hoffmann

News



- 11 **Perovskites Boost Solar-Cell Potential**
New materials could allow cheaper, more efficient solar cells for both traditional and novel applications.
By Don Monroe
-
- 14 **Gaming Machine Learning**
Game simulations are driving improvements in machine learning for autonomous vehicles and other devices.
By Samuel Greengard
-
- 17 **Parallel Computational Thinking**
Applications must be programmed to process instructions in parallel to take full advantage of the new multicore processors.
By Keith Kirkpatrick

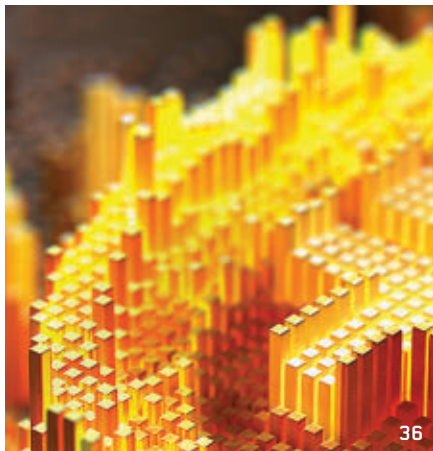
Viewpoints



- 20 **The Profession of IT**
The Forgotten Engineer
Engineering has been marginalized by the unhealthy belief that engineering is the application of science.
By Peter J. Denning
-
- 24 **Broadening Participation**
Community Colleges: A Resource for Increasing Equity and Inclusion in Computer Science Education
Challenging a simplistic pathway metaphor.
By Louise Ann Lyon and Jill Denner
-
- 27 **Kode Vicious**
Cold, Hard Cache
On the implementation and maintenance of caches.
By George V. Neville-Neil
-
- 29 **Viewpoint**
The Death of Big Software
We are past the tipping point in the transition away from 20th-century big software architectures.
By Stephen J. Andriole
-
- 33 **Viewpoint**
Lousy Advice to the Lovelorn
The 37% rule is rarely applicable in real-world situations. It is certainly entirely wrong-headed as advice for getting married.
By Ernest Davis



Practice



36 **Bitcoin's Academic Pedigree**
The concept of cryptocurrencies is built from forgotten ideas in research literature.
By Arvind Narayanan and Jeremy Clark

46 **XML and JSON Are Like Cardboard**
Cardboard surrounds and protects stuff as it crosses boundaries.
By Pat Helland

48 **Research for Practice: Vigorous Public Debates in Academic Computer Science**
Expert-curated guides to the best of CS research.
By John Regehr

Q Articles' development led by **acmqueue**
queue.acm.org

Contributed Articles



52 **Cybersecurity, Nuclear Security, Alan Turing, and Illogical Logic**
Cyber deterrence, like nuclear deterrence, depends on our adversaries being rational enough to be deterred by our threats but us not by theirs.
By Martin E. Hellman



Watch the author discuss his work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/cybersecurity-nuclear-security-alan-turing-and-illogical-logic>

60 **Technology-Driven Changes in Work and Employment**
Even when surrounded by ubiquitous computing, humans should be encouraged to do what they do better than machines.
By Ramiro Montealegre and Wayne F. Cascio

About the Cover:



Martin E. Hellman, co-recipient of the 2016 ACM A.M. Turing Award, delivers his Turing Lecture (p. 52) as an absorbing, personal story weaving past and present, logic and illogic, cybersecurity and nuclear security, and even love and marriage. His story begins over 40 years ago—an era that inspired our cover image—just as the first crypto war was taking shape. Cover illustration by Shotopop.

Review Articles

68 **Energy Efficiency: A New Concern for Application Software Developers**
Development of energy-efficient software is hindered by a lack of knowledge and a lack of tools.
By Gustavo Pinto and Fernando Castor



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/energy-efficiency-a-new-concern-for-application-software-developers>

Research Highlights

78 **Technical Perspective Pricing Information (and Its Implications)**
By Aaron Roth

79 **A Theory of Pricing Private Data**
By Chao Li, Daniel Yang Li, Jerome Miklau, and Dan Suciu

87 **Technical Perspective A Simple, Elegant Approach to Non-Numeric Parallelization**
By James Larus

88 **Automatically Accelerating Non-Numerical Programs by Architecture-Compiler Co-Design**
By Simone Campanoni, Kevin Brownell, Svilen Kanev, Timothy M. Jones, Gu-Yeon Wei, and David Brooks



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

- Bobby Schnabel
- 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 Cappo
- Director, Office of Publications**
Scott E. Delman

ACM COUNCIL

- President**
Vicki L. Hanson
- Vice-President**
Cherri M. Pancake
- Secretary/Treasurer**
Elizabeth Churchill
- Past President**
Alexander L. Wolf
- Chair, SGB Board**
Jeanna Matthews
- Co-Chairs, Publications Board**
Jack Davidson and Joseph Konstan
- Members-at-Large**
Gabriele Anderst-Kotis; Susan Dumais; Elizabeth D. Mynatt; Pamela Samuelson; Eugene H. Spafford
- SGB Council Representatives**
Paul Beame; Jenna Neefe Matthews; Barbara Boucher Owens

BOARD CHAIRS

- Education Board**
Mehran Sahami and Jane Chu Prey
- Practitioners Board**
Terry Coatta and Stephen Ibaraki

REGIONAL COUNCIL CHAIRS

- ACM Europe Council**
Dame Professor Wendy Hall
- ACM India Council**
Srinivas Padmanabhuni
- ACM China Council**
Jianguang Sun

PUBLICATIONS BOARD

- Co-Chairs**
Jack Davidson; Joseph Konstan
- Board Members**
Phoebe Ayers; Karin K. Breitman; Terry J. Coatta; Anne Condon; Nikil Dutt; Roch Guerrin; Chris Hankin; Carol Hutchins; Yannis Ioannidis; Michael L. Nelsion; M. Tamer Ozsu; Eugene H. Spafford; Stephen N. Spencer; Alex Wade; Keith Webster; Julie R. Williamson

ACM U.S. Public Policy Office

1701 Pennsylvania Ave NW, Suite 300,
Washington, DC 20006 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Deborah Seehorn,
Interim 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
Rights and Permissions
Deborah Cotton
Editorial Assistant
Jade Morris

Art Director
Andriy Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Production Manager
Bernadette Shade
Advertising Sales Account Manager
Ilia Rodriguez

Columnists
David Anderson; Phillip G. Armour;
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>

AUTHOR GUIDELINES
<http://cacm.acm.org/about-communications/author-center>

ACM ADVERTISING DEPARTMENT
2 Penn Plaza, Suite 701, New York, NY
10121-0701
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)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF
Andrew A. Chien
aic@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
William Pulleyblank and Marc Snir
Board Members
Monica Divitini; Mei Kobayashi;
Michael Mitzenmacher; Rajeev Rastogi;
François Sillion

VIEWPOINTS Co-Chairs
Tim Finin; Susanne E. Hambrusch;
John Leslie King; Paul Rosenbloom
Board Members
Stefan Bechtold; Michael L. Best;
Judith Bishop; Mark Guzdial;
Richard Ladner; Carl Landwehr;
Beng Chin Ooi; Loren Terveen;
Marshall Van Alstyne; Jeannette Wing

PRACTICE Chair
Stephen Bourne and Theo Schlossnagle
Board Members
Eric Allman; Samy Bahra; Peter Bailis;
Terry Coatta; Stuart Feldman; Nicole Forsgren;
Camille Fournier; Benjamin Fried;
Pat Hanrahan; 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;
Elisa Bertino; Gilles Brassard; Kim Bruce;
Alan Bundy; Peter Buneman; Carl Gutwin;
Yannis Ioannidis; Gal A. Kaminka;
Ashish Kapoor; Kristin Lauter; Igor Markov;
Bernhard Nebel; Lionel M. Ni; Adrian Perrig;
Marie-Christine Rousset; Krishan Sabnani;
Ron Shamir; Alex Smola; Josep Torrellas;
Michael Vitale; Hannes Werthner;
Reinhard Wilhelm

RESEARCH HIGHLIGHTS Co-Chairs
Azer Bestavros and Gregory Morrisett
Board Members
Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Michael Backes; Maria-Florina Balcan;
Andrei Broder; Doug Burger; Stuart K. Card;
Jeff Chase; Jon Crowcroft; Alexei Efros;
Alon Halevy; Sven Koenig; Steve Marschner;
Tim Roughgarden; Guy Steele, Jr.;
Margaret H. Wright; Nikolai Zeldovich;
Andreas Zeller

WEB Chair
James Landay
Board Members
Marti Hearst; Jason I. Hong;
Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2017 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, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER
Please send address changes to *Communications of the ACM*
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Printed in the U.S.A.



Association for Computing Machinery





Andrew A. Chien

DOI:10.1145/3156284

Computing Is the Secret Ingredient (well, not so secret)

PERHAPS YOU REMEMBER the iconic theme of the globally popular Kung Fu Panda movies, “You are the secret ingredient!” This meant that self-belief is important and with it great things can be achieved—Po, for example, became the Dragon Warrior. My meaning here is that computer science is both a powerful enabler of rapid advances in all intellectual fields and a disruptor driving furious revolutions in commerce and society worldwide. Computer science is more important and potent than ever!

Computing is driving unprecedented rapid change. One cluster of revolutions is around artificial intelligence (AI) and machine learning. Every day brings evidence of the rapidly growing capabilities in AI, driven by a host of algorithmic advances, but notably machine learning, to perform tasks heretofore exclusively the province of humankind. With high-profile applications such as speech, question and answer systems, image and face recognition, robotics with growing autonomy and flexibility—including self-driving cars, deep ocean exploration, and space exploration—society is broadly aware of AI’s growing capabilities. And worldwide, industry is ablaze in virtually every sector with the specter of disruptive, radical new opportunities.

Further, as AI capabilities push computing into new domains, there is growing concern² in economic, policy, and computing communities about the potential impact on employment, types of work, and global competition.

A second cluster of revolution is around blockchain and decentralized trust. The advent of Bitcoin has produced a cascade of academic research, startup experimentation, industrial innovation,

and now an explosion of change. An initial burst of innovation around electronic currencies that improve Bitcoin to create currencies with different properties continues to make waves, sparking new activity, government regulation, and in some cases outright bans. Over time, innovation has matured and broadened into a staggering breadth of applications based on the core disruptions that decentralized trust represents. Reformulated business and government activities around a distributed, trusted ledger abound, including new approaches to provenance, decentralized markets, and a host of financial applications, but few examples of the structural disruptions from centralized to distributed trust that blockchain is driving.

Equally exciting are efforts to recast foundational Internet services such as DNS on blockchain ideas. The Internet was conceived as the decentralized network of peers, but the design of core services, such as DNS, have always depended on trusted services—and organizations. Reinventing these services’ decentralized trust presents a radical new future for these and perhaps many types of Internet services and distributed systems.

While these two clusters of revolutions are enabled by generations of Moore’s Law and large-scale distributed systems, its notable that the essence of each is algorithmic advances and breakthroughs. Algorithms are in the essential core of the field of computer science.

A third revolution is the growing recognition that computer science is a fundamental element of secondary education. Recent signs include formal adoption of computer science education guidelines for primary and secondary education in the U.K.,³ an ACM-Industry vision,¹ and a \$200M U.S. Department of Education program¹ to

promote high-quality STEM and computer science education curriculum and programs in high schools.

While these are just three, there are doubtless many more, and I would love to hear about them!

Computing is more important than ever and driving disruptive change. But it’s also moving faster than ever because of so many creative and innovative computing professionals, huge capabilities in chips and massive clouds, and companies with extraordinary scope and ability to drive innovation. One of the giants of our field, the late Jim Gray, said in his 1999 Turing lecture, “Boy, we thought computing was moving fast in the 70’s and 80’s, but it’s really moving fast now.” I believe the rate of change that computer science is advancing more rapidly and driving more change in society today ... and with dramatically broader scope.

So, sit back for a moment these holidays as a computing professional and revel in where we are, and think about the exciting opportunities—and responsibilities—we have before us!

Andrew A. Chien, EDITOR-IN-CHIEF

Andrew A. Chien is the William Eckhardt Distinguished Service Professor in the Department of Computer Science at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Laboratory.

References

1. ACM and Partners Release Framework for Computer Science Education in U.S. K–12 Schools, Oct. 18, 2016.
2. De Lange Conference on Humans, Machines, and the Future of Work, Dec. 5, 2016.
3. United Kingdom Department of Education. Statutory Guidance: National Curriculum in England: Computing Programmes of Study, Sept. 11, 2013.
4. U.S.A. Presidential Memorandum on Creating Pathways to Jobs by Increasing Access to High-Quality Science, Technology, Engineering, and Mathematics and Computer Science Education. Sept. 25, 2017.

Copyright held by author.

Start CS Students Off with Industry Best Practices

LAMENTING THAT CS students are often not exposed to best practices in the classroom, software engineer Thomas A. Limoncelli offered advice for serving students better in his article “Four Ways to Make CS and IT More Immersive” (Oct. 2017). We agree with that sentiment, as we reported in our Viewpoint “Crossing the Software Education Chasm” (May 2012), describing development of a course very much in line with Limoncelli’s recommendations. We continue to welcome instructors to use our course materials and approach for Engineering Software as a Service, or ESaaS, originally developed at the University of California, Berkeley, as a way to follow Limoncelli’s guidelines:

Use best-of-breed dev/ops tools from the start. In our ESaaS course, students use Git from day one, including (in their open-ended design project) badges for coverage, CodeClimate, and continuous integration.

Homework programs, even Hello World, should generate Web pages, not text. This requires some minimum understanding of SaaS architecture, languages, and moving parts. ESaaS’s week-two assignment requires that students modify and deploy to the public cloud a simple SaaS app built with the Sinatra framework. The goal is to get them “thinking SaaS” early on by addressing several questions: How is state persisted? How does the app interact with the user? How do routes and URIs map to actions the user wants to take? And how can apps be structured as a set of RESTful resources and operations?

Curricula should start with a working system that reflects best practices, not just build from low-level to higher-level abstractions. The same assignment requires that students examine the code of a simple Sinatra app that demonstrates good coding practices, including integration and unit tests—before being introduced to creating or

reading code for such tests.

Focus on reading, understanding, and adding value to existing systems, a practice much more common in software engineering than greenfield development. ESaaS includes a two-part homework assignment on enhancing legacy code, but, more valuable for students, an increasing fraction of available team projects for nonprofits undertaken in Berkeley’s version of the course—10 of 11 projects in Summer 2017 and 13 of 20 projects in Fall 2017—are functioning legacy systems in need of additional features. Agile-Ventures, a U.K. nonprofit with which we work closely and whose developer-training programs closely follow the ESaaS pedagogy and methodology, also curates a portfolio of such “legacy systems” to support its mission of training developers to contribute to open-source projects in agile teams, as do several other colleges using our materials.

We applaud Limoncelli for urging CS instructors to more closely track best practices in industry. We invite students and instructors to try our free two-part (soon three-part) MOOC sequence—called Agile Development Using Ruby on Rails—on edX and invite instructors to avail themselves of the wealth of instructor materials available through <http://www.saas-book.info/>, including auto-graders to relieve them of having to manually grade these realistic assignments.

Armando Fox and **David Patterson**,
Berkeley, CA, and
Sam Joseph, Harrow, U.K.

Inspire Ethical Behavior Within the Profession

In his Editor’s Letter “Computing Is a Profession” (Oct. 2017), Editor In Chief Andrew A. Chien said those in computing should welcome, educate, and mentor new generations, not just as programmers but as professionals. We agree. That is why, in 2010, we and

15 other individuals established The Pledge of the Computing Professional (<http://computing-professional.org/>) to recognize graduates of computing programs as professionals in service to society, as the Order of the Engineer (<http://www.order-of-the-engineer.org/>) does with graduates of U.S. engineering programs. Today, 38 institutions in the U.S. conduct the Pledge’s rite-of-passage ceremony as part of their graduation activities. Graduates taking the Pledge sign a certificate both publicly and in the presence of their peers and are then presented with a pin to remind them of their commitment to self-accountability through ethical and moral behavior within the profession.

The Pledge has been endorsed by the Order of the Engineer, the ACM Special Interest Group on Computers and Society, and the ACM Committee on Professional Ethics. We would welcome a more comprehensive collaboration with ACM toward our mutual goal of promoting the attributes of deep technical expertise, essential, valued, societal contribution, and the need to adhere to high ethical and technical standards characteristic of the noblest ambitions of the profession.

For more on the Pledge, see <http://computing-professional.org/> or contact us directly.

John K. Estell, Ada, OH, and
Ken Christensen, Tampa, FL

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.



Vinton G. Cerf

DOI:10.1145/3154767

Now for Something Entirely Different

DEPARTING FROM MY usual stream of consciousness, this column is about three books I have just read: *Bullsh*t*,^a *Future Babble*^b and *Deep Future*.^c The first two get at the proliferation of wrong but persuasive assertions about the past, present, or future. The last one appeals to logic and humility. I draw them to your attention because I found them usefully thought-provoking and often very clarifying.

In *Bullsh*t*, John Grant systematically demolishes a wide range of mistaken beliefs and illustrates human foibles that often lead us to believe the unbelievable because we want to, not because the arguments for somehow outweigh the arguments against. In a world filled with misinformation (whether intentional or out of ignorance), disinformation, and scientific theories that have been falsified by new experimental evidence, we need all the tools we can muster to put claims to rigorous test. This takes real work and even some pain as some favorite notion is undermined by counterevidence. Good science demands that we be prepared to abandon long-held beliefs when confronted by new facts. Once it was thought that neutrinos had no mass, now we find they have very small, variable mass and the various flavors of neutrinos oscillate from one flavor to another while traveling from their origins. Current evidence makes extremely lightweight

objects even weirder than they were when they were first predicted to account for conservation of mass/energy in subatomic interactions. While Grant does not deal with neutrinos, he does cope with endless examples of “junk science, bogus claims, wacky theories, and general human stupidity” to quote the subtitle of his book.

In *Future Babble*, Dan Gardner goes to great lengths to explain the dynamics and even fundamental aspects of human nature that lead us to accept predictions that prove to be wrong. He explains Paul Ehrlich’s elevation and recognition (that is, many prizes and awards) for his *Population Bomb*^d book and the subsequent failure of most of his predictions to materialize. Gardner presents examples of the rationalizations that lead people to cling to favored theories and beliefs. He distinguishes “hedgehogs” from “foxes” in that the hedgehog knows only one thing and is certain of it (and conveys this conviction emphatically) and the foxes know they don’t know everything and are prepared to cope with discovering error and adapting to it. We learn what it is the hedgehogs project that induces some to believe them and not others who have humility in the face of unknown unknowns.^e We crave certainty and predictability and uncertainty makes us uncomfortable. Even our brains try hard to find patterns in noise to make sense of the world around us.

Both of these books should be required reading for people struggling to

filter good information from bad in all sources, but especially in our increasingly online world.

In *Deep Future*, Curt Stager, a climatologist, lays out the consequences of global warming, citing credible reasons for human contribution to increased greenhouse gases that trap heat in the atmosphere. The author’s most interesting observations takes us 55 million years into the past when the so-called Paleocene-Eocene Thermal Maximum (PETM)^f produced a warming period lasting about 200,000 years after which Earth was returned to its previously scheduled ice age. Stager uses a significant body of scientific evidence to show how increased atmospheric carbon content produced measurable and significant increases in average temperature and increased ocean acidity with consequences for flora and fauna. A surprise for me was his observation that the warming we have apparently launched may actually postpone the next ice age (predicted to come in about 50,000 years based on detectable cycles) for up to 400,000 years. He points out that ice ages may be far more damaging to human society than global warming despite the predicted and very negative side effects of the latter.

If you read any of these, let me know what you think. ■

^f https://en.wikipedia.org/wiki/Paleocene%E2%80%93Eocene_Thermal_Maximum

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

Copyright held by author.

^a *Bullsh*t*, J. Grant (pseudonym of Paul Barnett), MJF Books, NY, 2014.

^b *Future Babble*, D. Gardner, Penguin Group, NY, 2011.

^c *Deep Future: The Next 100,000 Years on Earth*, C. Stager, St. Martin’s Press, 2011.

^d *The Population Bomb*, P. Ehrlich, Buccaneer Books, NY, 1971.

^e Whatever else you may think of Donald Rumsfeld, his explication of “unknown unknowns” is creditable.

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.

twitter

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

DOI:10.1145/3148245

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

Building Tools to Help Students Learn to Program

Philip Guo summarizes his first three years of research into building tools to support those learning computer programming.



Philip Guo
Learning Programming at Scale

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

August 8, 2017

My current research trajectory centers on what I call learning programming at scale. Decades of prior research have worked to improve how computer programming is taught in traditional K–12 and university classrooms, but the vast majority of people around the world—children in low-income areas, working adults with full-time jobs, the fast-growing population of older adults, and millions in developing countries—do not have access to high-quality classroom learning environments. Thus, the central question that drives my research is: *How can we better understand the millions of people from diverse backgrounds who are now learning programming online and then design scalable software to support their learning goals?* In my first three years as an assistant professor so far, I have approached this question from three main directions:

1. Understanding why and how people from diverse backgrounds are learning programming.

One critical prerequisite for improving how programming is taught is to understand why and how people are currently learning and what obstacles they face. To work toward this goal, I have been studying traditionally under-represented learner populations and non-traditional learning environments.

I recently studied how older adults aged 60 and over are learning programming.¹ I found that they were often motivated by age-related reasons such as keeping their brains challenged as they aged, making up for missed learning opportunities during their youth, connecting with younger family members, and improving their job prospects. They reported a variety of age-related frustrations such as a perceived decline in cognitive abilities, lack of social opportunities to interact with tutors and peers, and trouble dealing with constantly changing software technologies.

With Chris Parnin and his students, we studied the unique challenges faced by female programmers when seeking and providing help on the popular Stack Overflow question-and-answer website.⁸ We found five participation barriers that affected women more than men: 1) not being aware of certain features of the site, 2) not feeling qualified enough to chime in with questions and answers, 3) being intimidated by the large size of the online community, 4) discomfort from interacting with strangers online, and 5) fear of appearing like they are slacking on the job.

My student Jeremy Warner and I also studied the recent phenomenon of *hackathons* where college students gather for 24-to-36-hour periods to learn coding by creating software prototypes.³ We found that the time-limited format of hackathons generated excitement and focus and that learning occurred incidentally, opportunistically, and from peers. However, some students were discouraged from attending by perceptions of an overly competitive climate, an unwelcoming culture, and fears of not having enough prior experience.

Parmit Chilana and I identified and studied an emerging population of college students¹⁰ and professionals at technology companies⁹ who want to learn programming but do not actually need to write code for their jobs. We call these people *conversational programmers* since their main goal is to learn just enough about programming to be able to hold productive technical conversations with programmers.

2. Designing new kinds of programming environments to support learners.

Current programming environments are designed to maximize the productivity of professionals who are already experts. Instead, I've been creating new environments to address the unique challenges faced by novice programmers, which can hopefully ease their path to eventually becoming experts.

First, I built a series of tools to help novices overcome a fundamental barrier to learning programming: understanding what happens “under the hood” as the computer runs each line of source code. These tools are all built on top of the Python Tutor Web-based programming environment (<http://pythontutor.com/>) that I created in 2010.¹⁶ Python Tutor (despite its outdated name!) lets users write code in languages such as Python, Java, C, C++, JavaScript, TypeScript, and Ruby; it runs the user's code and automatically visualizes what the computer is doing step-by-step. So far, over 3.5 million people from over 180 countries have used Python Tutor to understand and debug their code.

I extended Python Tutor with a real-time collaborative mode called Codechella¹², which lets multiple users connect to the same visualization session and work together to solve programming problems and tutor one another. I followed up on these ideas with Codeopticon¹¹, a real-time activity monitoring dashboard that allows a single tutor to simultaneously watch dozens of people working on the Python Tutor website and jump in to tutor multiple learners at once. My student Hyeonsu Kang and I then morphed Python Tutor into Omnicode,⁶ a live programming environment that continually visualizes the entire history of all program values to give programmers a bird's-eye view of execution. With Chris Parnin and our student Ian Drosos, we created HappyFace,⁷ a medically inspired pain scale embedded into Python Tutor to let users self-report their frustration levels.

The second set of challenges that I tackle here relates to the fact that novices have trouble installing, configuring, and managing the complex array of software tools required to become productive as programmers. I coined

a term called “*command-line BS-ery*” to refer to these sources of extrinsic complexity that demoralize novices. To help remove these complexities, Jeremy Warner and I created CodePilot,² a programming environment that lets novices quickly get started with pair programming and test-driven development by integrating real-time collaborative coding, testing, bug reporting, and version control management into a single unified system. Similarly, my student Xiong Zhang and I created DS.js,⁵ which lets novices get started learning data science by writing code to analyze data directly on any webpage instead of needing to download datasets and configure analysis software on their own computers.

3. Designing new formats for programming-related instructional materials.

My third major research direction involves studying the shortcomings of existing formats for programming-related instructional materials and then designing new instructional formats that improve the user experience for both creators and consumers of those materials.

My students and I analyzed all of the discussion forum messages in a popular programming MOOC¹⁴ and also how people navigated a computer programming digital textbook.¹⁵ We found that people often wanted to discuss runtime code execution state but had lots of trouble doing so since forums are purely text-based. From these findings, we propose that a better discussion forum for learning programming should integrate automatically generated visualizations of execution state and enable inline annotations of source code and output.

My student Mitchell Gordon and I created Codepourri,¹³ a new tutorial format and crowdsourcing workflow that lets Python Tutor users work together to create step-by-step tutorials by directly annotating runtime code visualizations. Since there are far more learners than experts, using learners as a volunteer crowd of workers is a potentially more scalable way to create coding tutorials than relying solely on experts. We found that crowd-created tutorials for simple code were accu-

rate, informative, and even contained insights that expert instructors missed.

Back to eliminating command-line BS-ery, my student Alok Mysore and I created Torta⁴, a macOS app that allows users to create step-by-step tutorials that span multiple command-line and GUI applications by simply demonstrating the intended actions on their computers. The Torta system: a) automatically records a screencast video along with relevant operating system events, b) generates a new kind of mixed-media tutorial with the benefits of both video and text formats, and c) gives step-by-step feedback to people who are following the tutorial and automatically runs certain steps. ■

References

- Guo, P.J. Older Adults Learning Computer Programming: Motivations, Frustrations, and Design Opportunities, *CHI 2017* (Honorable Mention Paper Award)
- Warner, J., and Guo, P.J. CodePilot: Scaffolding End-to-End Collaborative Software Development for Novice Programmers, *CHI 2017*.
- Warner, J., and Guo, P.J. Hack.edu: Examining How College Hackathons Are Perceived By Student Attendees and Non-Attendees, *ICER 2017*.
- Mysore, A., and Guo, P.J. Torta: Generating Mixed-Media GUI and Command-Line App Tutorials Using Operating-System-Wide Activity Tracing, *UIST 2017*.
- Zhang, X., and Guo, P.J. DS.js: Turn Any Webpage into an Example-Centric Live Programming Environment for Learning Data Science, *UIST 2017* (Honorable Mention Paper Award)
- Kang, H., and Guo, P.J. Omnicode: A Novice-Oriented Live Programming Environment with Always-On Run-Time Value Visualizations, *UIST 2017*.
- Drosos, I., Guo, P.J., and Parnin, C. HappyFace: Identifying and Predicting Frustrating Obstacles for Learning Programming at Scale, *VL/HCC 2017*.
- Ford, D., Smith, J., Guo, P.J., and Parnin, C. Paradise Unplugged: Identifying Barriers for Female Participation on Stack Overflow, *FSE 2016*.
- Chilana, P.K., Singh, R., and Guo, P.J. Understanding Conversational Programmers: A Perspective from the Software Industry, *CHI 2016*.
- Chilana, P.K., Alcock, C., Dembla, S., Ho, A., Hurst, A., Armstrong, B., and Guo, P.J. Perceptions of Non-CS Majors in Intro Programming: The Rise of the Conversational Programmer, *VL/HCC 2015*.
- Guo, P.J. Codeopticon: Real-Time, One-To-Many Human Tutoring for Computer Programming, *UIST 2015*.
- Guo, P.J., White, J., and Zanelatto, R. Codechella: Multi-User Program Visualizations for Real-Time Tutoring and Collaborative Learning, *VL/HCC 2015*.
- Gordon, M., and Guo, P.J. Codepourri: Creating Visual Coding Tutorials Using A Volunteer Crowd Of Learners, *VL/HCC 2015*.
- Zhu, J., Warner, J., Gordon, M., White, J., Zanelatto, R., and Guo, P.J. Toward a Domain-Specific Visual Discussion Forum for Learning Computer Programming: An Empirical Study of a Popular MOOC Forum, *VL/HCC 2015*.
- Warner, J., Doorenbos, J., Miller, B.N., and Guo, P.J. How High School, College, and Online Students Differentially Engage with an Interactive Digital Textbook, *EDM 2015*.
- Guo, P.J. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education, *SIGCSE 2013*.

Philip Guo is an assistant professor of cognitive science at the University of California, San Diego. His research spans human-computer interaction, online learning, and computing education. Learn more at <http://pgbovine.net/>

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 Abel Prize, the Fields Medal, and the Nevanlinna Prize.

The Heidelberg Laureate Forum is **September 23–28, 2018** 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?

New and recent Ph.Ds, doctoral candidates, other graduate students pursuing research, and undergraduate students with solid research experience and a commitment to computing research

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 process is open between **November 6, 2017** and **February 9, 2018**.

We reserve the right to close the application website early depending on the volume

Successful applicants will be notified by **mid April 2018**.

More information available on Heidelberg social media



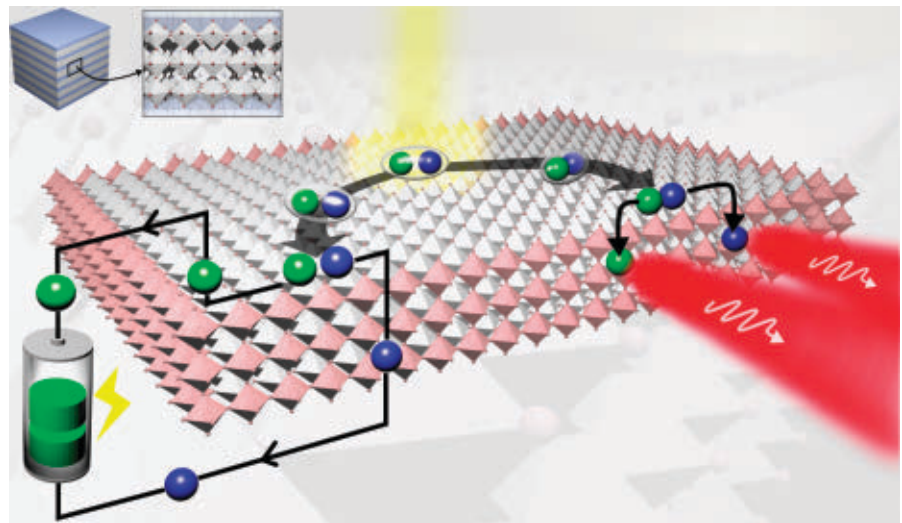
Perovskites Boost Solar-Cell Potential

New materials could allow cheaper, more efficient solar cells for both traditional and novel applications.

OVER THE PAST five years, rapid progress in photovoltaic technology has been further accelerated by materials called perovskites. They require only common ingredients and relatively easy manufacturing methods, holding out the possibility of cheap thin-film cells on a variety of surfaces or combined with silicon in large panels. In the laboratory, small-area cells made with these materials already feature solar-conversion efficiencies as high as 22%, rivaling those of traditional silicon solar cells.

“I’ve been in the business since the late 70s,” said David Cahen, a chemist and materials scientist at the Weizmann Institute in Rehovot, Israel. “This is a time of developments in solar cells and photovoltaics that is unprecedented,” including new materials, efficiency improvements, and cost reductions. But “there has not been anything like this one,” he said. “The perovskites have put all of those into shadow.”

A practical technology will require further engineering for large-scale manufacture and long-term stability, and there are still open scientific questions. Moreover, although researchers think they can avoid environmental



Material structure sketching the stacking of nanometer-thick layers of two-dimensional perovskite and organic spacing layers.

toxicity from lead in the materials, they may not be able to avoid a public perception of danger.

But perovskites’ potential is clearly shaking up the field.

A New Twist

Perovskites are a well-known family of materials whose crystals consist of a lattice of octahedra, each comprising six negative ions around a positive ion, with a second positive ion in the gap between

the octahedra. The octahedra can easily rotate while maintaining contact at their corners, giving perovskites unusual properties that have long been exploited in specialized applications. Lithium niobate, for example, acts both as a piezoelectric transducer between electrical and mechanical signals and in high-speed optical modulators for telecommunications. In the new materials, the arrangement follows these rules, but the ingredients are different.

The first hints of the solar-cell potential of perovskite semiconductors came from Japan in 2009, but researchers really took notice in 2012 with the report of cells having an efficiency of about 8%. In these hybrid organic-inorganic lead-halide perovskites, the second positive ion is generally a large organic (carbon-containing) molecule, while the octahedra consist of lead (sometimes tin) surrounded by halogen ions (from Group VII of the periodic table, commonly iodine).

“There are reasons why they escaped people’s attention,” said Michael McGehee, a professor of Materials Science and Engineering at Stanford University in Palo Alto, CA. “They have an inorganic aspect to them and then they have an organic ion in there as well.” The current workhorse solar-cell material is silicon, carved from large single-crystal wafers, although other inorganic materials are also employed. Organic semiconductors, similar to those in some LED displays, also make good solar cells, and can be deposited near room temperatures. But the hybrid perovskites are “very different” than either of these, McGehee said.

Importantly, hybrid perovskites encompass a whole class of materials, depending on which components are combined in the crystal framework. By combining different metal, halogen, and organic components, researchers can tune the material’s optical and electronic properties, as well as the chemical interactions that form the complete device. “There are many thousands of materials,” McGehee said, and, accounting for electrode and interface materials needed for a complete device, “millions of combinations.”

Harvesting Photons

Extraction of electrical energy from sunlight requires complex, highly engineered structures. Various layers are combined to first direct sunlight into the semiconductor layer and then to pass the generated electricity to the outside world, and to prevent water or air from reacting with the materials. But the central role involves a semiconductor property known as its bandgap, which is the energy range separating low-energy states that are initially filled with electrons from higher-energy empty states. A photon, or quantum of light, that has more energy than the bandgap can liber-

ate a negatively charged electron into the empty “band,” leaving behind a positively charged “hole.”

If the electron and hole make their way to opposite electrodes connected to the material, they provide that energy to the outside world as an electric current. The voltage at which this current is supplied is limited by the bandgap (measured in electron-volts). If the photon has more energy than this, the excess is lost as heat, while photons with too little energy to span the bandgap simply pass through without being absorbed. Because sunlight includes photons with a wide range of energies, the electrical power output of a single cell cannot exceed about 33% of the light energy.

However, the actual efficiency is always lower than this theoretical limit. One reason is that an electron can refill an empty state, locally “recombining” with a hole, rather than traversing the external circuit. This process occurs much faster when there are electronic states in the bandgap, an energy range that would ideally be empty. A key feature of the hybrid-perovskite materials is that they seem to have few electrically active defects. The resulting long lifetime before recombination, together with speedy motion of electrons and holes, makes it easier to approach the ideal efficiency.

Cahen suspects that defects are rare in perovskites because of their soft, floppy crystal structure, which he regards as “dynamically disordered,” and possibly even “self-healing.” McGehee, by contrast, said that “the material is full of defects,” but that the defects are not a problem because of the chemical

By combining different metal, halogen, and organic components, researchers can tune the material’s optical and electronic properties.

structure of the material. In many semiconductors, he noted, one can regard the bandgap as reflecting the energy separation of “bonding” and “anti-bonding” combinations of states on neighboring atoms, so broken bonds at a defect create states with energy in the middle of the bandgap, where they are most troublesome. In the perovskites, by contrast, McGehee says the gap has a different origin, and defect states do not promote recombination.

Whatever the theoretical explanation, experimenters have already pushed the efficiency of a single perovskite cell above 22%, compared to almost 27% achieved for the best crystalline silicon cells after decades of work.

Cool Manufacturing

Beyond their high efficiency, hybrid perovskites have other useful features: their ingredients are abundant, and thin films can be formed near room temperature by either liquid- or vapor-based processing. These properties should make them inexpensive to create compared to silicon crystals grown at very high temperatures. Still, Henry Snaith, a physicist at the University of Oxford, cautions that quality and uniformity are at least as important as equipment cost for large-scale production.

Low-temperature processing also allows deposition of thin-film photovoltaics on surfaces that could not tolerate high heat. For example, emerging applications known as building-integrated photovoltaics (BIPV) encompass solar cells deposited directly on roof shingles or on the window glass of an office tower, where they generate power while letting some light through.

These applications inspired Snaith to start a company called Oxford Photovoltaics in 2010 to develop dye-sensitized organic photovoltaics, which use one molecule to absorb the light and others to carry the electrical current (and which can also be deposited at low temperatures). Yet after promising results from his own university group and others, the company “rapidly shifted all activity towards perovskites.” The company also shifted from the “tiny” BIPV market to mainstream solar panel applications, Snaith said. “We recognized very early on that perovskites could augment silicon in a tandem device,” taking advantage of the low-temperature processing

to fabricate a perovskite cell on top of an existing crystalline-silicon cell.

Tandem cells can extract more energy from the solar spectrum, potentially beating the efficiency limit for any single cell. By using a material with a larger bandgap, the perovskite cell makes better use of the high-energy part of the solar spectrum, while lower-energy photons continue into the silicon cell to generate a voltage more closely matched to their energy. Published perovskite/silicon tandem-cell efficiencies are as high as 23.6% (about 26% when the cells are wired separately). An alternative approach, combining two perovskite materials with different bandgaps, has demonstrated efficiency of 20.3% in the separately wired configuration.

Scaling Up

In spite of the rapid research progress in small perovskite-based solar cells, much more work will be needed for a commercial technology. Practical modules require large, uniform, high-quality films, for example, as well as ancillary circuits to keep individual cells from hampering the entire array. Toward this end, Snaith said, Oxford Photovoltaics has acquired a facility in Brandenburg, Germany, and is aiming toward the end of 2018 for pilot production on six-inch-diameter silicon wafers. Other companies may not be far behind.

Because the perovskites can react chemically with water or air, commercial cells will probably need multiple levels of protection or encapsulation. Customers will expect extra assurance that panels using these new materials will be stable over multiple years of use.

Researchers are also still exploring which ingredients make the best semiconductor (not to mention the various interface, electrode, and encapsulation materials). “If I had to make a prediction, I would say the material has not been found,” McGehee said. Snaith noted his company has settled on a particular composition for its initial manufacturing, but he expects that will change over the years.

One component that has proven difficult to change is lead, which has a well-deserved nasty reputation because of insidious health impacts from past use in gasoline and paint. “We haven’t been able to do anything that remotely approximates these materials

Researchers are still exploring which ingredients make the best semiconductors (not to mention the various interface, electrode, and encapsulation materials).

without lead,” Cahen said. (Researchers have had some success by replacing lead with tin, but these materials have had stability issues so far.)

Because the perovskite layer need only be less than a micron thick, however, compared to hundreds of microns of silicon, the overall fraction of lead in a module is quite small. In addition, any encapsulation required for stability should keep the lead in place. “Although it’s something we’re very sensitive to,” Snaith said, “I think there’s very little chance of any real environmental risk, provided panels are sensibly deployed and recycled at the end of their lifetime.” He acknowledged, however that public perception could still be an issue.

“It does mean I don’t think you’ll see this in portable products,” McGehee said. For panels, he speculated, 30% efficiency of cost-effective tandem designs is “very do-able,” he speculated, whereas silicon panels max out at 22%–23%. **Q**

Further Reading

Shackly, W., and Queisser, H.J. Detailed Balance Limit of Efficiency of p-n Junction Solar Cells, *J. Appl. Phys.* 32, 510 (1961). <http://dx.doi.org/10.1063/1.1736034>

Bush, K.A., et al. 23.6%-efficient monolithic perovskite/silicon tandem solar cells with improved stability, *Nat. Energy* 2, 17009 (2017). <http://dx.doi.org/10.1038/nenergy.2017.9>

Perovskite Cells for Tandem Applications, *École Polytechnique Fédérale de Lausanne* <http://pvlab.epfl.ch/page-124775-en.html>

Don Monroe is a science and technology writer based in Boston, MA.

© 2017 ACM 0001-0782/17/12 \$15.00

ACM Member News

HELPING GOVERNMENT ACCELERATE INNOVATION



Technology entrepreneur Robert H Zakon has over 20 years’ experience as a consultant,

applying the latest innovations in computing technologies to address client challenges. His experience ranges from engineering the tallest Wi-Fi link in the northeastern U.S. (atop Mount Washington, in New Hampshire), to working on cybersecurity policy for the healthcare records of hundreds of millions of people, to designing and developing one of the first corporate intranets.

Regarding his career trajectory, Zakon says, “I am usually the odd person out. I work on whatever is interesting and challenging.”

Born in Brazil, Zakon was 12 years old when he was given the opportunity to come to the U.S. He completed high school in New Jersey, then earned both his undergraduate and master’s degrees in computer engineering from Case Western Reserve University in Ohio.

Most recently, Zakon served as a White House Presidential Innovation Fellow, participating in a program designed to foster innovation and bring start-up sensibilities to the federal government.

“I was looking for an opportunity to do some work for the public good, but could not make a long-term commitment,” Zakon says. The Presidential Innovation Fellowship, which lasted a year, gave Zakon the opportunity to work at a policy level; he partnered with leaders at the U.S. Treasury, Department of Defense, and the Office of Science and Technology Policy, to try to accelerate innovation.

“The interaction between industry, academia and government is invaluable,” Zakon said, “and I would encourage others to take a turn at public service.”

—John Delaney

Gaming Machine Learning

Game simulations are driving improvements in machine learning for autonomous vehicles and other devices.

OVER THE LAST few years, the quest to build fully autonomous vehicles has shifted into high gear. Yet, despite huge advances in both the sensors and artificial intelligence (AI) required to operate these cars, one thing has so far proved elusive: developing algorithms that can accurately and consistently identify objects, movements, and road conditions. As Mathew Monfort, a postdoctoral associate and researcher at the Massachusetts Institute of Technology (MIT) puts it: “An autonomous vehicle must actually function in the real world. However, it’s extremely difficult and expensive to drive actual cars around to collect all the data necessary to make the technology completely reliable and safe.”

All of this is leading researchers down a different path: the use of game simulations and machine learning to build better algorithms and smarter vehicles. By compressing months or years of driving into minutes or even seconds, it is possible to learn how to better react to the unknown, the unexpected, and unforeseen, whether it is a stop sign obscured by graffiti, a worn or missing lane marking, or snow covering the road and obscuring everything.

“A human could analyze a situation and adapt quickly. But an autonomous vehicle that doesn’t detect something correctly could produce a result ranging from annoying to catastrophic,” explains Julian Togelius, associate professor of computer science and engineering at New York University (NYU).

The use of computer games and simulations—including the likes of open-source *TORCS* (The Open Racing Car Simulator) and commercially available *Grand Theft Auto V*—already is revolutionizing the way researchers develop autonomous vehicles, as well as robots, drones, and other machine systems.



A scene from Rockstar Games' *Grand Theft Auto V*, which is helping to revolutionize how researchers develop autonomous vehicles.

Not only is it possible to better understand machine behavior—including how sensors view and read the surrounding environment—it offers insights into human behavior in different situations. “These games offer extremely rich environments that allow you to drive through a broad range of road conditions that would be difficult to duplicate in the physical world,” says Artur Filipowicz, a recent graduate in operations research and financial engineering at Princeton University who has used machine learning to advance research on autonomous vehicles.

The Road Less Traveled

Although the idea of using video game simulations and AI to boost real-world performance for autonomous vehicles has been around for more than a decade, the concept has zoomed forward over the last few years. The rise of graphics processing units (GPUs) and the advent of convolutional neural net-

works (CNNs) suddenly made it possible to explore scenes and scenarios in deeper and broader ways. By tossing vast numbers of images at the artificial neural network—stop signs, traffic signals, road markings, barriers, trees, dogs, pedestrians, other vehicles, and much more—and comparing actions and reactions such as steering, braking, and acceleration, it’s possible to cycle rapidly through an array of events and scenarios en route to more refined algorithms and better performing self-driving cars.

Of course, the allure of this approach is that in the virtual world, cars never run out of fuel or need new tires, and they’re able to log millions of miles in a single day. There are no fatigued drivers and no risk of real-world collisions or injuries. However, the benefits don’t stop there.

“One can say that the real world is richer in terms of character than the virtual world, but in the virtual world

you can create specific situations and scenarios and study them faster and better,” says Alain Kornhauser, professor of operations research and financial engineering, and director of the Transportation Program, at Princeton University. “The big advantage is that you can focus in on specific ‘corner cases,’ the really difficult situations that represent the greatest risk and lead to the greatest number of crashes.”

About three years ago, Chenyi Chen, then a Ph.D. candidate at Princeton and now a deep learning researcher for autonomous driving at NVIDIA, began exploring the concept in earnest. He turned to the open-source car racing game TORCS to supply low-resolution visual data for a deep learning network. Working with Kornhauser, they devised a method for grabbing still images from the game and plugging them into the CNN. Chen then studied how to train a network for highway driving and how to judge the distance of other vehicles using 12 hours of human driving within the video game. “We realized we could create any situation we wanted and recreate any trajectory we desired. The game images provided a way to study driving in difficult situations, including rain, sleet, hail, and snow,” Kornhauser explains.

All of this attracted attention in the AI and autonomous vehicle communities. For example, Filipowicz decided to study stop signs to understand how humans recognize and react to signs. “Distance is difficult to measure in the real world but easy to measure in the virtual world, even under adverse weather conditions,” he explains. Filipowicz tapped *Grand Theft Auto V* for its rich and highly varied environment; it includes more than 250 models of vehicles, thousands of pedestrians, and animals, along with realistic settings and weather conditions. Future simulations might focus on additional detection, including signs covered with dirt, faded, partially obscured by fog, trees branches, or other objects, and those completely obscured by paint or graffiti, or broken off entirely. “The performance on both real and synthetic data, while not perfect, are promising,” he says.

Researchers at Darmstadt University of Technology in Germany and Intel Labs have also turned to *Grand Theft Auto V* to develop and fine-tune algo-

“The game images provided a way to study driving in difficult situations, including rain, sleet, hail, and snow.”

rithms that could be used by auto manufacturers, while a China-based startup electric auto manufacturer, NIO, has turned to simulations to design and build a fully autonomous vehicle that it hopes to bring to market in 2020. In recent months, Waymo, the autonomous vehicle arm of Alphabet (Google’s parent company), has begun using simulators to study every situation and variation engineers can imagine, including multiple vehicles changing lanes at the same time in close proximity, and the car recognizing road debris that could damage a vehicle or pose a crash hazard.

Yet, while the use of video games and AI has already caught the eye of major automotive companies, putting the data to full use is not without challenges. Transforming pixels and RGB values into useful data for a vehicle or other machine is a steep challenge, Kornhauser says.

In addition, NYU’s Togelius, who has experimented with computer games and AI to better understand player performance, as well as events and systems within the game, says the virtual and physical worlds do not always mesh neatly. What’s more, not all game scenarios are faithful to physical reality. In some cases, “It’s possible to learn from a simulation, take the knowledge into the real world and then find out that things don’t comply to the simulation. So, it is necessary to take a very iterative approach to AI and video game simulations and carefully validate results.”

AI in Overdrive

The use of AI and images to drive real-world gains shows no signs of subsiding. For instance, Monfort and a group of researchers at NVIDIA have used a CNN to map raw pixels from time-

Milestones

Computer Scientists Receive MacArthur ‘Genius’ Grants

Two computer scientists were among the 24 people selected to receive John D. and Catherine T. MacArthur Foundation fellowships.

The MacArthur Fellowship is a \$625,000, no-strings-attached award intended to encourage people of outstanding talent to pursue their own creative, intellectual, and professional inclinations.

Regina Barzilay is a computer scientist in the Department of Electrical Engineering and Computer Science of the Massachusetts Institute of Technology.

Barzilay is a computational linguist developing machine learning methods that enable computers to interpret unstructured document content and perform real-world tasks with the promise for significant societal impact.

Barzilay has made significant contributions to a wide range of problems in computational linguistics, including both interpretation and generation of human language.

Stefan Savage is a computer scientist in Department of Computer Science and Engineering of the University of California, San Diego.

Savage uses an interdisciplinary approach to address challenges to computer security and to counter cybercrime. In addition to identifying technological deficiencies, he contextualizes cybersecurity threats within much broader ecosystems, including underlying economic incentives and social structures contributing to vulnerabilities.

Savage has created new strategies for defending against malware and distributed denial of service attacks. He and colleagues measured network-level interactions to characterize the value chain of Internet-related crime.

Said Cecilia Conrad, managing director of the MacArthur Fellows Program, “These new MacArthur Fellows bring their exceptional creativity to diverse people, places, and social challenges. Their work gives us reason for optimism and inspires us all.”

—CACM Staff

stamped video captured by a single front-facing camera. With minimum training data from humans, the neural net learned to drive in traffic on local roads and highways with or without lane markings or guardrails by using the underlying mathematics of human steering angles as the training signal. What's more, the project accomplished the task across a wide spectrum of road and weather conditions in less than 100 hours. It also learned to operate in areas with unclear visual guidance, such as in parking lots and on unpaved roads. Monfort describes the method, which led to a test on an actual vehicle, as "surprisingly powerful."

Although Monfort's research involved actual video and real-world data rather than game images, it demonstrated the promise of deep learning applied to game graphics and actual videos—and how the two are closely related. For one thing, deep learning used for both synthetic and actual images could eliminate the need for a near infinite number of "if...then...else" statements, which are impractical to code when dealing with the randomness of the road. For another, this type of data could be combined with game data using a generative adversarial network (GAN), which relies on two neural networks "competing" with one another to boost machine learning. This approach could bridge the gap between data collected from the synthetic world and data from the physical world, Filipowicz says. "This may be the next phase of research. You could transfer the learning from one network to the other using either simulated or real-world images," he explains.

In addition, researchers are increasingly attracted to video games and simulations to understand how to build better robots, drones, and agents. By applying the same type of deep learning techniques, they can discover things that would have previously gone undetected. For example, in 2015, Microsoft embarked on a project called Malmo, which created an AI-based development platform revolving around the popular world-building game *Minecraft*. The goal of the project was to experiment with and study complex virtual environments and apply the lessons learned

Researchers are increasingly attracted to video games and simulations to understand how to build better robots, drones, and agents.


from that study to the physical world. Katja Hofmann, chief researcher for the project, has stated that "endless possibilities for experimentation" exist. Others, such as Google's DeepMind project, are also examining games and how they can apply data to the physical world.

Togelius says the goal is to build smarter systems and agents that can continue to tap data and adapt on the fly. Within this framework, humans could learn from agents, agents could learn from humans, and agents could learn from other agents. He says a "competitive, co-evolutionary process" could result in neural nets and learning systems that are better adapted to the increasingly blurry line between silicon and carbon-based intelligence. Within games, they would "handle more of the unexpected features of the physical world," but also fuel real-world gains by finding relationships and correlations that humans probably would not or could not notice.

Not surprisingly, there are limitations to how videogames can be used to train robotic and autonomous systems. Software such as *Grand Theft Auto V* typically requires hundreds of millions of dollars to develop, yet these packages are available commercially at a relatively low cost. Essentially, the game manufacturer is footing the bill for research and development that would be unachievable and unaffordable in a lab. As a result, the use of games for machine learning will likely be limited to specific fields, such as autonomous vehicle and robotics research. It's difficult to envision a game for training surgical robots, for example.

Nevertheless, the idea of using AI to

extract data from games and apply it to the real world is gaining momentum. Not only do these simulations eliminate the cost, time, and human resources involved with building and operating complex machines—autonomous vehicles, robots, drones, software agents and more—they make it possible to cycle through millions of possibilities and find subtle anomalies and correlations that determine whether an autonomous vehicle maneuvers correctly for a dog on the road and stops at a traffic light that is not working, or simply crashes.

Says Kornhauser, "Humans are very good at recognizing situations. In order to build autonomous vehicles and other devices that work correctly, we must understand and translate all the factors and variables to a machine. It's a challenge that AI can solve." 

Further Reading

Bainbridge, L.

Ironies of automation. New Technology and Human Error, J. Rasmussen, K. Duncan, J. Leplat (Eds.). Wiley, Chichester, U.K., 1987, 271–283.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K.

End to End Learning for Self-Driving Cars, April 25, 2016

<https://arxiv.org/pdf/1604.07316v1.pdf>

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. **DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving**, *Proceedings of 15th IEEE International Conference on Computer Vision (ICCV2015)*, May 2015

<http://deepdriving.cs.princeton.edu>

Loiacono, D., Lanzi, P.L., Togelius, J., Onieva, E., Pelta, D.A., Butz, M.V., Lönneker, T.D., Cardamone, L., Perez, D., Sáez, Y., Preuss, M., and Quadflieg, J.

The 2009 Simulated Car Racing Championship, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 2, June 2010

<http://julian.togelius.com/Loiacono2010The.pdf>

Filipowicz, A.

Virtual Environments as Driving Schools for Deep Learning Vision-Based Sensors in Self-Driving Cars, June 2017.

http://orfe.princeton.edu/~alaink/Theses/SeniorTheses'17/Artur_Filipowicz_VirtualEnvironmentsAsDrivingSchools.pdf

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

Parallel Computational Thinking

Applications must be programmed to process instructions in parallel to take full advantage of the new multicore processors.

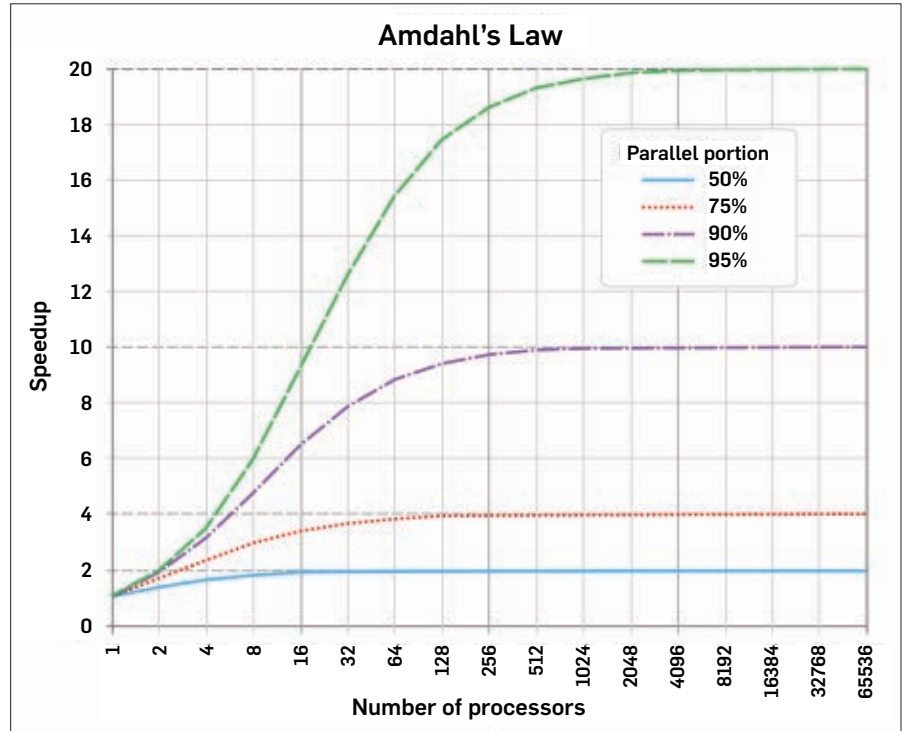
WHEN LEARNING A new skill, it is often advantageous to start out simply, and then incorporate greater complexity as the learner gains greater experience, expertise, and familiarity with the subject at hand.

Indeed, most computer science education has followed that line of thinking, teaching beginning computer science students to write programs that perform one instruction at a time, and then move on to the next instruction. This is known as sequential programming, and it has largely been the accepted model of computer science instruction at both the university and K-12 levels, in contrast with parallel computing, a model of programming where multiple instructions are processed simultaneously.

“The educational system is, mostly through inertia, still focused on the computing paradigm of the 20th century, which was one processor executing instructions one after another, so algorithmic problem-solving was mainly oriented toward a sequential model,” explains Charles (Chip) Weems, an associate professor of computer science at the University of Massachusetts.

Today, however, nearly all applications running on smartphones, tablets, and PCs, are powered by multicore processors, which are necessary when working with the large datasets that drive both consumer applications, such as a Twitter or Facebook feed, as well as business and commerce-related applications, such as travel deal sites, weather applications, and real-time traffic data. To take full advantage of these multicore processors, programming applications to process instructions in parallel—which allows multiple instructions to be processed at the same time—is required.

Teaching new computer science



A chart illustrating Amdahl's Law, which says the speed-up of a program from parallelization is limited by how much of the program can be parallelized.

students to think and program in parallel will not only better prepare them to code and program these devices, but also helps to train their minds to think in abstractions to solve problems, rather than simply in terms of writing code.

“Computer architectures have been doing parallelism at the instruction level for decades in a way that the vast majority of programmers can ignore that it's there,” says Dan Grossman, a professor at the Paul G. Allen School of Computer Science & Engineering at the University of Washington, and a member of the ACM steering committee on computing curricula, which concluded its work in 2013. “If that had remained the only form of parallelism, there would be a much weaker argument for teaching parallelism at the undergraduate level. But that has not remained the dominant form; it

has run its course. We have added other forms of parallelism that do not hide the issue as much for programmers.”

Computing societies have recognized the need to incorporate parallelism as part of a core collegiate computer science curriculum. The ACM and the IEEE jointly introduced new guidelines in 2013, and recommended integrating parallel education throughout the curriculum. Although these are only guidelines, and most universities still tend to teach parallel programming concepts only to more advanced students, there is a growing push to incorporate parallelism in college-level programming courses from the start.

“In the last 15 years, systems have gone almost entirely parallel,” Weems says. “Unless you're talking about small embedded systems, everything

you might encounter is a multicore that's multithreaded, and nearly everything comes with a graphics processor, which can be programmed."

Moreover, programming in parallel—and training students to think at a higher level—allows for programming to be more direct and concise, compared with sequential programming, according to Guy Blelloch, a professor and Associate Dean for Undergraduate Programs in the computer science department at Carnegie Mellon University. "It's not so much that parallel code is simpler than sequential code, it's that in the abstraction of code [it] basically makes [the problem to solved] simpler and at the same time makes it parallel."

Blelloch, who notes that CMU teaches parallelism from the start in its Intro to Data Structures and Algorithms Course during the first semester of sophomore year, says that by teaching beginning computer science students to think about problems in terms of abstractions, they are able to move beyond a basic understanding of programming to get the heart of the matter of actually solving problems.

"There's a lot of emphasis on intro programming on a loop," Blelloch says. "And that's really not that interesting. Often you just want to be thinking, 'I want to add five to every element in this array.' I could start the loop at the beginning, but the right way to think

When a student learns parallelism, it "helps them develop a more flexible approach to problem solving because there are more algorithmic models to draw upon."

of it is I just want to add five to every element in the array. By doing the parallelism, you're more focusing on the underlying ideas, as opposed to getting stuck in details of loops."

Another added benefit of parallelism, Weems says, is that when a student learns parallel programming, it "helps them develop a more flexible approach to problem solving because there are more algorithmic models to draw upon." While there's also the added benefit of learning how to break apart larger problems into simpler ones, parallelism also requires programmers to learn to see alternate abstractions of the problem.

"There are situations where a problem can be decomposed into subtasks, but various factors result

in still having to choose among algorithmic approaches," Weems says. "There are times when communicating via shared memory is most effective, while in other cases it's better to work locally and communicate via messages, or to use a combination of these approaches at different levels of granularity. [Parallelism] forces programmers to look more explicitly and holistically at the interactions that take place among the data and operations in solving the problem, by considering them from more perspectives than the sequential model."

Weems, a member of the working group for the Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER), which is funded by the U.S. National Science Foundation (NSF), also notes that there is a demand for new programmers who have parallel programming skills, from government science labs as well as large technology industry companies.

Some major universities, including the University of Massachusetts, where Weems is a faculty member, have begun to incorporate parallelism into their curricula, with promising results. For example, a faculty-authored paper from Texas State University highlights the success of its new curriculum, which was launched in the 2016–2017 academic year. According to the paper, parallel

Technology

Out of Sight, Into Mind

French researchers are developing a video technology that will completely bypass the eyes and project an image directly into one's brain.

Essentially, they want to enable the blind to see again, without ever having to rely on the human eye.

The researchers have been partially successful in their quest; so far, they have found a way to model how the human retina captures visual information using machine vision, a camera, and a computer.

"We aim at extending this modeling to the visual cortex" of the brain, says Serge Picaud, scientific supervisor at Institut de

la Vision in Paris, France, who is working with Jose Alain Sahel, the Institut's director, to return sight to those who live in darkness.

Both Picaud and Sahel are part of a larger, \$21.6-million initiative overseen by the U.S. Defense Advanced Research Projects Agency (DARPA), which is looking to use technology to enable the blind to see, the deaf to hear, and the speech-impaired to talk.

Picaud and Sahel hope their nascent technology will ultimately work by using a specially designed machine vision camera from Chronocam, which will feed imagery from the outside world into a pocket

computer. Once processed, the visual signal will be broadcast to wireless devices the researchers plan to implant inside the brain, which in turn will fire individual neurons in the brain's visual cortex to create sight.

"The implanted devices containing LED arrays will deliver light stimuli on the visual cortex," Sahel says.

Picaud and Sahel will be experimenting on animals before they move into the human brain.

Says Phillip Alvelda, manager of DARPA's Neural Engineering System Design (NESD) program, "By increasing the capacity of advanced neural interfaces to

engage more than one million neurons in parallel, NESD aims to enable rich, two-way communication with the brain at a scale that will help deepen our understanding of that organ's underlying biology, complexity, and function."

"Its deeper complexities are going to remain a mystery for some time to come," Alvelda adds, "but if we're successful in delivering rich sensory signals directly to the brain, NESD will lay a broad foundation for new neurological therapies."

—Joe Dysart is an Internet speaker and business consultant based in Manhattan.

computing concepts are introduced and reiterated via a series of short, self-contained modules across several lower division courses. Then, most concepts are combined into a senior-level capstone course in multicore programming. The evaluations conducted during the first year displayed encouraging results for the early-and-often approach in terms of learning outcomes, student interest, and confidence gains in computer science.

Still, some educators are not convinced that introducing parallelism during introductory or lower-division computer science courses is necessary in order to produce well-trained computer programmers of the future.

“We live in a world of multicore devices,” says Mark Guzdial, a professor in the School of Interactive Computing at the Georgia Institute of Technology. “But I don’t know if we should be teaching [parallel programming] to everyone. It may be better off to start with sequential programming, and then move on to parallel.”

Whether to teach elements of parallelism in early coursework may also depend on the focus of the coursework, Grossman says, noting that the key to integrating parallelism is to limit the complexity of the program itself, by ensuring that computations and variables are not highly dependent upon one another during parallel processing operations.

“The way to get parallelism to work correctly is to have as few shared variables as you can that can change,” Grossman says, noting instead of setting up a single variable that may change its value, it may make more sense to simply program a second variable that can hold the second value.

“There are different ways to teach introductory programming without parallelism that make it harder or easier to add parallelism later,” Grossman adds. “For initial exposure to programming for younger, pre-college students, I haven’t seen much focus on parallelism and I think that’s fine.”

Hansel Lynn owns Silicon Valley-based theCoderSchool, an afterschool coding instruction franchise that works exclusively with children ages 8 to 18. Lynn believes sequential coding should be taught first. “For kids aged 8–18, we always teach sequential coding first,” Lynn says. “Especially for

“We live in a world of multicore devices, but I don’t know if we should be teaching [parallel programming] to everyone. It may be better off to start with sequential programming.”

younger kids, sequential, logical stepping is an important foundation for learning how to think logically.”

Lynn does note that some platforms may provide some exposure to parallel programming. “At times however, platforms such as Scratch do allow kids to get some exposure to parallel programming (such as multiple objects moving and detecting collision simultaneously),” Lynn says. “While we may create projects with parallel programming concepts embedded, we don’t typically focus on parallel thinking, as we find it dilutes the focus on building their sequential logic thinking skills.”

Whether at the collegiate or secondary level, there are challenges related to revamping the curriculum to include parallelism. First, many professors have not had been exposed to parallelism on a programming level, particularly if they were educated before parallel processing became mainstream, which occurred about a decade or so ago. There are also non-technical issues, such as getting buy-in from other faculty members, as well as the challenge of updating online tutorials and auto-graders, which must be revamped to deal with different types of code. Additionally, textbooks need to be augmented or amended, as many introductory texts don’t cover parallelism. “There was one that mentioned concurrency, but it was in Chapter 23,” Weems quips.

Blelloch says that the trend, for now, is to simply add a discussion about parallelism to existing coursework. “I think most departments are taking a some-

what more conservative approach,” he says. “They’ve taught this course for 20 years in a particular way and it’s not very hard to add three weeks at the end which makes them think in parallel.”

Perhaps the larger question for computer science educators revolves around selecting the right material to introduce to beginning computer science students.

“It’s very tempting in computer science education to think that we do students a service if we introduce ‘x’ from day one, for various values of ‘x’, and you’re asking about ‘x’ being parallelism,” Grossman says. “I do see value in that, but I also see value in introducing security from day one. I see value in introducing ethics from day one. I also see the value of introducing performance from day one. But there’s only one day one.

“Everyone who crafts a curriculum has to make choices about what they introduce from the beginning as the default way to think about a program, compared with what they push off until later,” Grossman says. “Trade-offs are trade-offs, and people can spend their lives studying pedagogy.”

Further Reading

Grossman, M., Aziz, M., Chi, H., Tibrewal, A., Imam, S., and Sarkar, V. Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level, *Journal of Parallel and Distributed Computing*, Volume 105 Issue C, July 2017, pp. 18-30
<http://dl.acm.org/citation.cfm?id=3085740>

Burtscher, M., Peng, W., Qasem, A., Shi, H., Tamir, D., and Thiry, H. Integrating Parallel Computing into the Undergraduate Curriculum at Texas State University: Experiences from the First Year
<http://bit.ly/2tl7rpe>

Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science The Joint Task Force on Computing Curricular: Association for Computing Machinery IEEE Computer Society December 20, 2013
<http://www.acm.org/education/CS2013-final-report.pdf>

Video:

Parallel Computing Explained:
<https://www.youtube.com/watch?v=q7sgzDH1cR8>

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

© 2017 ACM 0001-0782/17/12 \$15.00



DOI:10.1145/3152912

Peter J. Denning

The Profession of IT

The Forgotten Engineer

Engineering has been marginalized by the unhealthy belief that engineering is the application of science.

WE LIVE IN a time that reveres science. It was not always this way: in much of the previous centuries, engineers were heroes. In the late 20th century, however, the engineer's image eroded because science seemed to offer more hope with difficult problems and because technology seemed to inflict collateral damage through such issues as pollution, exploitation of nature, weapons of mass destruction, and massive surveillance.

Our modern fascination for science is marginalizing engineering. This is especially bad for computer science and engineering. For instance, we routinely teach programming as a set of abstractions to be applied rather than a skill of design to satisfy customers. We routinely make claims about what computing can theoretically accomplish without knowing that we can deliver.

Not long ago, *Science* magazine distributed a subscription solicitation that offered a T-shirt bearing on the front the image of a Leonardo da Vinci flying machine and on the back the inscription "Aviation. Brought to you by

science." This slogan was an XL misrepresentation of how aviation came to be.⁸

The linear model of research and development behind this slogan has been repeatedly challenged and disproved. *Science* advertised this popular fallacy on a T-shirt—with the worst example imaginable.

The Wright brothers did look to science for help in answering fundamental questions about wings and propellers.⁸ The premier institution of the day, the Smithsonian, was unable to help them. These bicycle mechanics and self-taught engineers conducted their own experiments, spending many hours studying the flights of birds to understand what enabled them to soar, eventually concluding that wing warping would be a key to controlling gliders and powered aircraft. Only when they had a sketch of a mechanical concept could they begin to apply science to its development. *Science* did not bring us aviation. Rather, the Wright brothers built working flying machines that opened the possibility of aviation and gave birth to a new science, aeronautics.

Much the same happened with computing technology. The first digital electronic computer was built in Germany in 1938 by Konrad Zuse, who was educated as a civil engineer. John Atanasoff and Clifford Berry built a digital computer that solved linear equations and demonstrated it in 1942. The first digital computer capable of any computable function—ENIAC (1945)—was built by electrical engineers J. Presper Eckert and John Mauchly under a U.S. Army contract. They spent many hours tinkering to find reliable electronic logic circuits. In 1945, they joined with Herman Goldstine, Arthur Burks, and John von Neumann to design a stored-program machine, which they demonstrated would be more powerful and significantly less complex than ENIAC.

Although Alan Turing, whom many computer scientists revere, proposed his Turing machine model of computation in 1936, his work was known only to a handful of mathematical logicians, and completely unknown to the engineers who built the first electronic computers.^{2,5} It was not until the 1950s,



when the first academic programs were being born, that Turing's work offered the theoretical basis to make computer science credible as a new department in universities. In other words, as important as Turing's work is, it did not inform or inspire the first electronic computers or the stored program concept. Instead, the success of the first stored-program electronic computers created the opening for Turing's work to become important. Yet we have our own T-shirts proclaiming that Turing was the father of digital computing.

Incongruities

Most engineering associations and the accrediting body ABET define engineering as "application of science and mathematics to finding practical solutions to problems," a definition that makes engineering an application of science or a branch of science. Not that long ago, some engineering associations defined engineering as the *science and art* of designing and making structures. Petroski comments, "Once that design is articulated by the engineer as artist, it must

be analyzed by the engineer as scientist in as rigorous an application of the scientific method that any scientist must make."⁷ What happened to the idea that engineering is *both* science and art?

When I recently reviewed the status of computational thinking in education,³ I noticed that the recommendations for curricula focus almost exclusively about the science-math side of computational thinking and have little or nothing to say about architecture or design side. This seemed odd because programs are designed to control machines, and moreover most of the new jobs in computing are in architecture and design. Why are architecture and design not showing on the computational thinking radars?

Software pioneer David Parnas has long been a critic of the adopted approach to teaching software engineering, which seemed to him to downplay an engineering view in favor of a math-theory view. He wants students to get plenty of practice programming and designing programs to meet customer requirements.⁶ He notes that the crux of software de-

velopment is "multiperson development of multiversion programs"—in other words, teams and organizations building families of software. Curricula that lack a strong emphasis on design cannot prepare their graduates for this. What happened to the engineering in software engineering?

Engineering has helped all the social, life, and physical sciences advance by providing tools and instruments. Engineering has helped civilization advance by providing reliable infrastructures such as electricity, transportation, water, and food. Since the 1980s, the engineering of supercomputers and networks has given birth to a raft of new branches of science, mostly called "computational X" where "X" names a traditional science. Most computing professionals today are heavily engaged in engineering—they design systems for customers and experiment to find out what works. Why has the engineering outlook lost favor in many CS departments?

Much engineering is oriented around design of systems and structures that will be reliable, safe, and se-

cure. Aside from “design of experiments and models,” scientists hardly ever discuss design. How could engineering be a subset of science when its main concern is not a concern of science?

Distinctions

It is clear that science and engineering are distinct enterprises with different ways of looking at the world. Yet they cannot advance without interacting with each other. Historians Bowler and Morus trace the evolution of the steam engine and the telegraph in the 1700s and 1800s.¹ They debunked the modern myths of one-directional flow from science to technology. In those days, there was no practical difference between science and technology. It seems that the distinction between science and engineering is recent—introduced in the late 1940s when Vannevar Bush advocated the establishment of the U.S. National Science Foundation for government support of basic research.

Given the contemporary definitions, I have found three distinctions between engineering and science

Design in computing is fundamentally an engineering practice.

particularly helpful. The first concerns the nature of their work. Engineers design and build technologies that serve useful purposes, whereas scientists search for laws explaining phenomena. Design is among the most common words of engineering, whereas it is uncommon in science. Design in engineering is a process of finding practical, safe, cost-effective implementations. Whereas scientists have a knack for finding recurrences, engineers have a knack for listening to clients and proposing technologies of value to them.

The second main distinction is how scientists and engineers regard knowledge. Scientists treat knowledge as data and information that have been organized into a “body of knowledge” that is then available for anyone to use. The scientific method is a process of standard, outside observers gathering and weighing evidence in support of claims that might be added to the body. Engineers treat knowledge as skillful practices that enable design and building of tools and technologies. Engineers are not disinterested outside observers; they are immersed in the communities of use. They embody practices for building, maintaining, and repairing technologies; attending to reliability, dependability, and safety in the context of use; and following engineering standards and codes of ethics.

The third main distinction concerns the role of abstractions and models. Science emphasizes models, and engineering machines. There is a fundamental distinction between modeling machines and building them. Abstractions are useful for what they leave out. Machines are useful for what they leave in. Hardware and software are interchangeable to the theorist, but not to the engineer.

The familiar phrase “devil is in the details” is an engineer’s motto. Engineers must get the details right for systems to work. Scientists want to eliminate the details so that the recurrences stand out.

The accompanying table summarizes, compares, and contrasts how computer scientists and engineers tend to view design. These are dispositions and tendencies, not formal definitions. Computer scientists need to function with both worldviews.

Engineering and Science in Computing

As we noted in 1989,⁴ science, engineering, and mathematics are irrevocably interwoven in the fabric of computing. Every computing technology has a science, an engineering, and a mathematics aspect. Computing cannot be dissected into the three components. It is not a branch of science, engineering, or mathematics. In computing, design means developing practical systems with the aid of mathematical tools such as program

Dispositions toward design in computing.

| Science | Engineering |
|--|---|
| A design is a plan or a blueprint for a model or an experiment | Design is a process of proposing systems that meet customer concerns |
| Designs aim to reveal causes | Designers aim to harness naturally occurring effects |
| Designers find and validate models | Designers align software with user practices |
| Designers work with proven abstractions and models that omit inessential details | Designers know that every detail counts for a reliable and safe product or system |
| Designers are ultimately concerned with whether claims are true | Designers are ultimately concerned with whether products or systems work |
| Designers are objective observers detached from communities | Designers are immersed in their communities |
| Designers aim to understand the world | Designers aim for working implementations that can change the world |
| Correctness and validation measure success | Client satisfaction measures success |
| Mistakes can be eliminated with formal verification | Mistakes and defects are inherent, the system must tolerate them |
| Good designs can be formally verified so that they will work the first time | Good designs are fault tolerant so that they continue to be reliable and safe even when faults and defects appear |
| Good designs rule out contingencies or surprises | Designers work with contingencies and surprises |
| Experiments validate hypotheses | Tinkering is experimenting to find what works |
| What we know is expressed as our body of knowledge | What we know is expressed in our practices, standards, and lore about what works |
| Engineering and technology will apply the science | We build technologies to have something to apply science to |

verifiers, practices from science such as taxonomies of design patterns, and validation methods such as careful statistical testing. Design in computing is fundamentally an engineering practice.

In computing, we work closely with the notion that programs can be expressed as structures of abstract objects, and the useful work happens when those abstract objects control machines that affect the world. The science-math mind plays a strong role with structuring the abstractions; the engineering mind plays a strong role with bringing the effects into the world. The field cannot survive if these two aspects do not maintain a synergized balance.

These arguments are not new. In his 1968 ACM A.M. Turing lecture, award recipient Richard Hamming argued that the computer is at the heart of computing; without it, almost everything computing professionals do would be idle speculation. In the past two decades, we have added natural information processes, such as DNA transcription, to what we study, but the computer remains the heart. Every programming language is a means for designers to control an abstract machine that when simulated produces useful and practical results. Computer science graduates, Hamming argued, must learn design in the context of bringing value to users.

This is why I am concerned that our academic departments embody too strong an emphasis on the theoretical side of computing. The engineering side has been diminished in the process. Recent reforms to computing curricula have introduced a new first course “CS principles.” Most of the content of these courses is concepts relating to programming and algorithm organization. A few departments, more so in engineering, use design courses and Raspberry Pi or Arduino labs to introduce students to the field. The teachers are always surprised by how much the students accomplish in the role of designers without much grounding in the science of the field. More departments ought to consider starting students with a design course.

The result is curricula that encapsulate computing inside a boundary of math-science-theory and diminish

crucial engineering aspects in architecture and design. This is unhealthy because most of the jobs for which our graduates are aiming are much more strongly oriented around engineering than science. It is no wonder that employers complain that CS graduates do not fit and need extensive training and hand holding to become profitable employees.

It bothers me that all the modern advances—in AI, machine learning, big data, cloud computing, and computer security—are touted as triumphs of science rather than what they really are, achievements of engineering and science working together.

Conclusion

Science and engineering need each other. Neither is the application or fulfillment of the other. Science emphasizes the discovery of recurrences. Engineering seeks to harness effects before the recurrences are fully known. Science moves in when the effect has proved useful and we seek to understand it better, optimize it, make it more reliable, and exploit its recurrences for prediction. Science takes care of abstractions, engineering the details that enable abstractions to work. The marriage of science and engineering in computing is critical for the continued health of the field. ▣

References

1. Bowler, P.J. and Morus, I. *Making Modern Science: An Historical Survey*. University of Chicago Press, 2010.
2. Daylight, E. A Turing tale. *Commun. ACM* 57, 10 (Sept. 2014), 36–38.
3. Denning, P. Remaining trouble spots with computational thinking. *Commun. ACM* 60, 6 (June 2017), 33–39.
4. Denning, P. et al. Computing as a discipline. *Commun. ACM* 32, 1 (Jan. 1989), 9–23.
5. Haigh, T. Actually, Turing did not invent the computer. *Commun. ACM* 57, 1 (Jan. 2014), 36–41.
6. Parnas, D. David Parnas speaks of software engineering. CCSL Centro de Competência em Software Livre, 2014; <http://ccsl.ime.usp.br/en/news/14/09/17/david-parnas-speaks-software-engineering-ccsl>
7. Petroski, H. *To Engineer is Human: The Role of Failure in Successful Design*. Vintage, 1992.
8. Petroski, H. and Denning, P. Your science T-shirt doesn't fly. *ACM Ubiquity* (Dec. 2016); <http://ubiquity.acm.org/blog/your-science-t-shirt-doesnt-fly/>

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, 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 U.S. federal government.

The author thanks Fernando Flores and Henry Petroski for conversations with insights about science and engineering.

Copyright held by author.

Calendar of Events

December 4–6

K-CAP 2017: Knowledge Capture Conference, Austin, TX, Sponsored: ACM/SIG, Contact: Oscar Corcho, Email: ocorcho@gmail.com

December 5–8

UCC '17: 10th International Conference on Utility and Cloud Computing, Austin, TX, Co-Sponsored: Other Societies, Contact: Alan Fraser Sill, Email: Alan.Sill@ttu.edu

December 11–15

Middleware '17: 18th International Middleware Conference, Las Vegas, NV, Sponsored: ACM/SIG, Contact: Anshul Gandhi, Email: anshul@cs.stonybrook.edu

December 12–15

CoNEXT '17: The 13th International Conference on Emerging Networking Experiments and Technologies, Incheon, Republic of Korea, Contact: Taekyoung Kwon, Email: tkkwon98@gmail.com

2018

January

January 7–10

GROUP '18: 2018 ACM Conference on Supporting Groupwork, Sanibel Island, FL, Sponsored: ACM/SIG, Contact: Michael Prilla, Email: prilla.michael@googlemail.com

February

February 21–24

SIGCSE '18: The 49th ACM Technical Symposium on Computing Science Education, Baltimore, MD, Sponsored: ACM/SIG, Contact: Tiffany Barnes, Email: tiffany.barnes@gmail.com

Broadening Participation Community Colleges: A Resource for Increasing Equity and Inclusion in Computer Science Education

Challenging a simplistic pathway metaphor.

IN THE LONG-STANDING and persistent lack of diversity in computer science (CS), too little attention has been paid to the role that community colleges (CCs) can play. Community colleges are poised to provide an important resource for preparing a 21st-century workforce begging for more computer science graduates as well as more diversity in those graduates. To broaden participation in CS, CCs must be the focus of increased study and intervention. As part of this work, we revisit the current metaphors that guide CS education research and practice—the idea that students follow a “pipeline” or even a “pathway” is too simplistic to capture the convoluted routes that many CC students are constrained to follow. In this column, we argue that there is a misalignment between the existing research and practice approaches and institutional structures, which are based on a traditional educational pipeline metaphor, and the experiences of students attempting to pursue a CS bachelor’s degree.

Community colleges are typically open access, lower-division institutions whose student population is more diverse than that of four-year universities and reflect a transforma-



tion of the demographics in higher education. As noted in an earlier *Communications* Broadening Participation column,⁸ there is a high participation of minorities in CS at CCs; more than half of CC students are non-white, and more than half of all Hispanic and Black undergraduates start at community college.¹ Efforts to retain

students through transfer to completion of a bachelor’s degree would be a large step forward in helping diversify the field.

Despite these differences between populations of students at CCs and at four-year institutions, research and institutional strategies continue to be informed by assumptions and

findings that do not always apply to students who begin their journey to a bachelor's degree at a CC. We argue that an understanding of the unique strengths and challenges of CC students is needed to strengthen efforts to broaden participation in CS. In addition, we claim that educational pathway metaphors need to be updated and applied to most effectively implement institutional changes that will encourage students to continue through CCs on to completion of bachelor's degrees. Specifically, we argue that efforts to increase equity and inclusion in CS should increase the focus on community colleges, and employ metaphors that more accurately represent the lived experiences of CC students.

Students Expect a Modified Traditional Education Pipeline Model

A traditional educational model—based on previous generations of white, upper-middle class men—follows a pipeline metaphor in which students prepare for professional, white-collar jobs by entering directly into and graduating from four-year postsecondary institutions (see “Traditional Education Model” diagram). As states have shifted the burden of paying for higher education to students, many are attracted to what they view as an equal but less-expensive model that includes two initial years at a community college before transferring and completing a bachelor's degree at a four-year institution in preparation for the workforce (see “Expected Education Model” diagram). In our research, we found that CS students chose community college and transfer over direct admission to a four-year institution often because they anticipated receiving the “same” degree while saving money.⁵ However, our work also reveals that the experienced pathways of students do not follow a traditional educational model; the fragility of the educational support structure for the CC student is such that the traditional education model is rarely attainable. In sum, when decision makers implement policies and procedures that rely on a pipeline model, they unwittingly suppress diversity.

When decision makers implement policies and procedures that rely on a pipeline model, they unwittingly suppress diversity.

Students Experience a Snarled Pathway Education Model

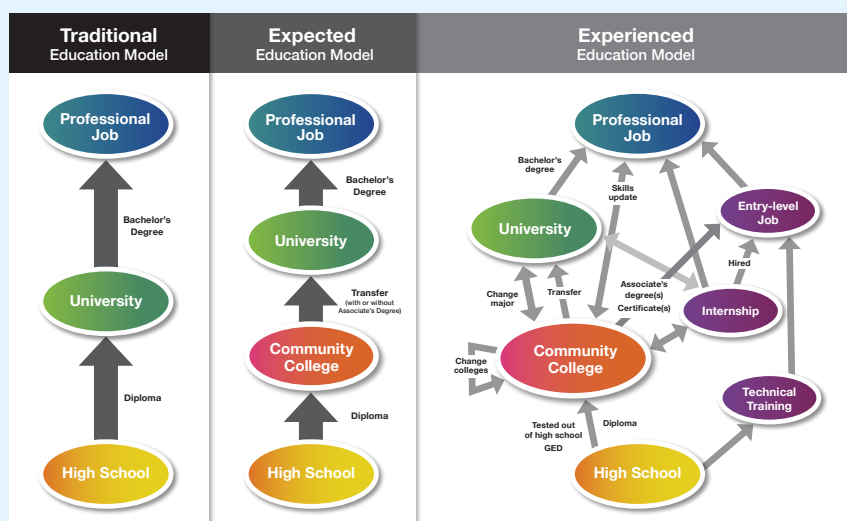
In contrast to the educational pipeline model that students are led to expect, community college students' pathways are often not direct or simple. For many students, they involve time off from school, enrollment, and reenrollment at different community colleges, and time in the workforce while also attending school. These twists and turns result in convoluted, individualized routes that can be full of detours and setbacks unexpected by students envisioning a simpler pipeline, and complicated by policies and restrictions at both CCs and four-year receiving institutions (see “Experienced Education Model” diagram). Jaggars et al.⁴ for example, found 1,213 distinct paths to graduation for CS bachelor's degree holders in their study.

Research provides support for what we argue is a more realistic model of CC students' experiences. For example, we found that students were delayed at CCs in preparation for transfer for many reasons including impacted CS classes, math anxiety or aversion, dropping and re-enrolling in classes in an effort to increase GPA, and family and financial responsibilities.⁵ Students struggled to create and follow a plan to transfer, complicated by the fact that receiving institutions had varying requirements for transfer admission. In addition to these difficulties—some of which are shared by all students at CCs—CS students had only vague ideas of how the field is applied in professional settings and how to prepare for a career in CS. For students who were five years out from taking an introductory programming class at a CC, pathways were complicated by incorrect advising, changing majors or schools, leaving school due to job responsibilities, and the necessity of starting their college years in remedial math classes.⁵

Suggestions for Simplifying CC CS Students' Pathways

Changes implemented to increase equity and inclusion in CS should be more responsive to the twists and turns of the pathways of CC students, while also working to straighten and simplify these pathways to more closely resemble a traditional education model. This includes increasing awareness about

Three education models.



the challenges students face and building structural supports to keep them moving forward. We advocate for several research-based strategies to be implemented at educational institutions.

Advising and Support. Studies have shown the importance of advising and support from faculty, counselors, and peers for the successful transfer from a CC to a four-year university in a STEM field⁷ and to help students remain on a CS pathway.⁵ But these sources of support should be based on an understanding of the complex and prolonged paths that CC students follow.

Culturally Responsive Practices. Faculty can play a critical role in ensuring policies and practices are tailored to respond to challenges faced by low-income students and those from underrepresented minority groups. In CS, this requires providing extra support and flexibility in schedules, developing clear CS transfer pathways, and building knowledge of CS careers.⁵

Gender Differences and Similarities. Strategies should also be responsive to gender differences and similarities. Studies show that while both women and men benefit from peer encouragement, CC women are more likely to intend to persist in CS if they are confident and interested in solving challenging problems, while men are more likely to intend to persist if they have positive interactions with their instructors and value computing.³ In addition, supports for men of color in CCs should be based on personal connections, engaged faculty and staff with high expectations, and opportunities to apply what they learn in the real world.²

Stronger Partnerships between Four-year Institutions and CCs. Four-year institutions should reach out to CCs to build true partnerships based on listening to CC faculty to make changes to streamline student pathways and increase flexibility along those pathways. Current patterns of transfer indicate the importance of articulation and transfer agreements⁶ and should be enhanced by faculty exchanges on advisory boards. At the University of Washington, Tacoma campus and at Kean University in New Jersey, such reciprocal visits have led

Research, policy, and interventions will be more effective if they are based on a realistic understanding of CC students' experiences.

to faculty's heightened awareness of impacts on transfer students of program decisions. Statewide websites such as <http://www.njtransfer.org> and <http://www.assist.org> help students have transfer information at their fingertips, but should be enhanced with individualized study plans to take into account the unique pathway of each student. To ease the transfer experience of CC students, efforts should be made to familiarize them with the settings, people, and events at receiving universities. Opening hackathons to CC student participation, university open houses or regional conferences that include CC students, direct mailings to CC students, and access to university counselors and research faculty are all best practices in this area.

Involvement from Industry. We also advocate for the involvement of the CS industry in the call to enable students—particularly from underrepresented groups—to have the resources and information they need to efficiently move through CCs to transfer, and then through four-year institutions to bachelor's degrees. Our research⁵ found that career knowledge early in CS postsecondary pathways gave students the impetus to persist in CS but that specific knowledge of job skills and preparation for particular jobs was lacking. Jaggars et al.⁴ found that students who grew up near a technology hub city were more

likely to attain a CS bachelor's degree; it is likely that these students were more familiar with the range of job opportunities in technology.

We argue that research, policy, and interventions will be more effective if they are based on a realistic understanding of CC students' experiences. We hope efforts to broaden participation in CS will increase their focus on CC students and faculty, and will lead to new, research-based interventions resulting in a marked increase in successful transfer and graduation and a broader diversity of students. **□**

References

1. American Association of Community Colleges. *Fact Sheet*, 2016; <http://bit.ly/2d3jnEI>
2. Center for Community College Student Engagement. *Aspirations to Achievement: Men of Color and Community Colleges*, 2014; <http://bit.ly/1mCjPrv>
3. Denner, J. et al. Community college men and women: A test of three widely held beliefs about who pursues computer science. *Community College Review*, 2014.
4. Jaggars, S.S. et al. *A Longitudinal Analysis of Community College Pathways to Computer Science Bachelor's Degrees*. Google Inc., Mountain View, CA, 2016; <http://bit.ly/2hOzEzW>
5. Lyon, L.A. and Denner, J. *Student Perspectives of Community College Pathways to Computer Science Bachelor's Degrees*. Google Inc., Mountain View, CA, 2016; <http://bit.ly/2yTajCO>
6. National Academies of Sciences, Engineering, and Medicine. *Barriers and Opportunities for 2-Year and 4-Year STEM Degrees: Systemic Change to Support Students' Diverse Pathways*. National Academies Press, 2016.
7. Packard, B.W. et al. Women's experiences in the STEM community college pathway. *Journal of Women and Minorities in Science and Engineering* 17, 2 (Feb. 2011), 129–147.
8. Taylor, V. and Ladner, R. Data trends on minorities and people with disabilities in computing. *Commun. ACM* 54, 12 (Dec. 2011), 34–37.

Louise Ann Lyon (LouAnn.Lyon@etr.org) is a Senior Research Associate at ETR (Education, Training, Research) in Silicon Valley, CA.

Jill Denner (jilld@etr.org) is a Senior Research Scientist at ETR in Silicon Valley, CA.

This material is based in part upon work supported by Google, Inc., and by the National Science Foundation under grant 0936791. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Google, Inc., or of the National Science Foundation.

Diagrams of educational models adapted from the Developing Computing Pathways presentation (<http://batec.org>). Thank you to Cheryl Calhoun of Santa Fe College, Pat Morreale of Kean University, and Cindy Tucker, and Melanie Williamson from Bluegrass Community, and Technical College for reviewing and providing comments on a draft of this column.



Kode Vicious Cold, Hard Cache

On the implementation and maintenance of caches.

Dear KV,

Our latest project at work requires a large number of slightly different software stacks to deploy within our cloud infrastructure. With modern hardware, I can test this deployment on a laptop. The problem I keep encountering is that our deployment system seems to secretly cache some of my files and settings and not clear them, even when I repeatedly issue the command to do so. I have resorted to repeatedly using the find command so that I can blow away the offending files. What I have found is that the system caches data in many places—not in one single, easy-to-find place—so I have started a list. All of which brings me to my question: Who writes this stuff?!

Out of Cache

Dear OOC,

I am not quite sure who writes this stuff, but I am pretty sure they do not know what they are doing. A wise person once said, “There are only two hard things in computer science: cache invalidation and naming things,” and that wise guy was supposedly Phil Karlton, but, no matter who said it, they are partially correct. Let’s skip the naming problem for now and go right into caching and the concerns one might have while maintaining a cache. You will note I said, “maintaining,” because plenty of people can build up a cache—that is just making copies of stuff you might want later and putting it in a place that is easier for you to find or faster for you to access. Maintaining a cache actually has several distinct operations, all of



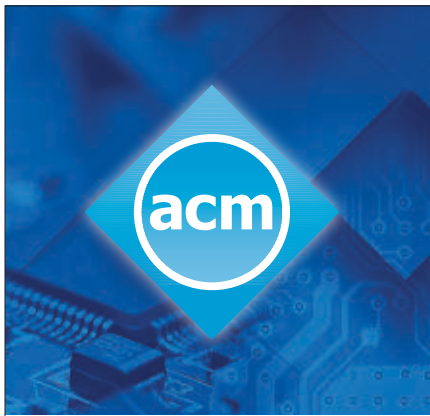
which must be carried out correctly for the cache to be of any real use.

The only reason to create a cache in the first place is that there is some piece of information that is frequently used and that can be stored in a place where it is quicker to access. For example, a system’s main memory can be thought of as a cache for files on disk, because it is perfectly possible to read a file into memory, execute the instructions read from the file, and then immediately free the memory. No one builds systems like that because it is expensive to keep copying the data from disk for each execution, so we leave the data we have copied in memory until we need the memory for some other purpose. Your particular case seems like a cache of either settings or files, rather than a memory, CPU, or chip-level cache, so I will narrow my remarks a bit more to cover just this case, but the principles apply to all sorts of caches.

Adding to a cache is relatively straightforward. As long as there is

room in the cache, an entry is added, hopefully, in a place that is quicker to access the second time around. Accessing an entry in the cache requires a lookup operation, which can fail. Failing to find an entry in the cache is called a “miss” and is the cache’s way of telling you to go back where you came from and look up the data in the old, slow way.

The problems you are having with your system come from its inability to invalidate stale data. If you change something in files that are being deployed and your deployment system does not notice that change, then it is going to think the data it has cached is fresh when, in fact, it is stale. Nothing stinks quite like stale data. Since I have no idea how your system is judging the freshness of your data, let’s just assume for a moment that it’s using the file modification timestamp, and then let’s further assume that it’s not looking at seconds, but instead, minutes. That means if you are doing a file save and then an immediate



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



A correctly implemented cache tracks all the things being cached.

“deploy,” your system will not, in fact, think that the data is stale. That is, if you do deploy-change-deploy in the same minute, which is quite possible for a touch typist, the system will not think that the old data is out of date. Another possibility is that the settings you’re changing are in a file that’s not being watched by the system, and that the thing that it cares about is some file that points to your file.

A correctly implemented cache tracks all the things being cached, not just the thing that the programmer assumes other people will modify. The problem usually comes as more types of data are added to the cache. A file here, a database entry there, and eventually what you have is not really a well-organized cache, but instead, the data-storage equivalent of spaghetti code, where files point to files, and the file with the dragon has the pellet with the poison, but the directory with the dragon has the file that is true.

One way for you, the user, to deal with this is to be a tad brutal and flush the cache. Whereas invalidation is the careful, even surgical, removal of stale data, a flush is just what it sounds like: it sends all the crap right down the drain, fresh *and* stale. It sounds like whoever implemented your system has made this difficult for you by scattering the requisite files all over the system, but once you have your complete list, it is probably time to write a flush.sh shell script to clear out all the craft you can think of.

Coming back up for air, let’s think about how a proper cache is managed, so that the next system any of us come across does not require the use of system-call tracing to find out why the system is misbehaving. Caches make sense only if they speed up access to frequently used data; otherwise, let the operating system or your database

handle things, because they are pretty smart at managing data and keeping the fresh data near at hand.

When you build a cache, pick a key that is easy to search for. Here is a hint: strings are not easy to search for, but hashes of strings are. When you add new object types to the cache, do not do it by adding a pointer from one object to another, unless those two objects must be treated as one thing. Trying to track down whether or not a sub-object has changed by walking all the parent objects is inefficient. When using timestamps to indicate freshness, look at the seconds—computers are fast; many things happen in a computer minute. If the cache contains files, put them all under a common root directory, as this will make flushing them easier for the user.

Much of what I have discussed is local caching, and KV has left out the distributed case, for reasons of personal sanity. The advice here is a good start to building a distributed cache, because if you get this wrong in the local case, your chances of getting it right in the distributed case are nil. If KV gets a good letter about the distributed case, he might take leave of his senses long enough to try to answer it, or he might burn the letter—and since sending letters is also a distributed system, there is no guarantee that you will know if the message was delivered, lost, or is in a stale cache.

KV

Related articles on queue.acm.org

What Are You Trying to Pull?

Kode Vicious

<http://queue.acm.org/detail.cfm?id=2931077>

Division of Labor in Embedded Systems

Ivan Godard

<http://queue.acm.org/detail.cfm?id=644266>

Beautiful Code Exists, If You Know Where to Look

Kode Vicious

<http://queue.acm.org/detail.cfm?id=1454458>

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.

Viewpoint

The Death of Big Software

We are past the tipping point in the transition away from 20th-century big software architectures.

WHY WOULD ANYONE undertake a multi-year software project today? Or upgrade an in-house-hosted legacy application? Or build—or use—anything that behaved like a monolithic software application? Big software project failure data is legendary.¹¹ There are myriad horror stories with titles like “9 VERY Scary ERP and ERP System Implementation Statistics.”¹² The Standish Group actually labels their annual technology project analyses as “Chaos Reports.”¹⁴ They reported that 66% of all technology projects completely or partially failed in 2015.

So assuming that management is reasonably well informed, it knows that big software projects are likely to fail. Yet in the 1990s and early 21st century there were still companies willing to try their hand with big software and prove they were unlike the others who failed so spectacularly. In spite of this unjustifiable optimism, many of these companies also failed. Even the U.S. Defense Department failed spectacularly.⁶

So that no one thinks that failure only plagues ERP applications, the data suggests all kinds of big software projects fail.¹³ Big customer relationship management (CRM) projects fail. Big database management systems (DBMS) projects fail. Big infrastructure projects fail. Big communications projects fail. In fact, most software projects designed to address enterprise-wide



problems with single, integrated platforms fail. Failure crosses vertical and functional areas as well, including retail, government, financial services, and even science.¹⁰

The high rate of failure helped kill big software. But there were other causes of death.

Causes of Death

Big software is dead. There were lots

of assassins. Some were all business, some were hiding in the governance trenches, some were up in the clouds and some were architectural. Let's look at the assassins in a little detail.

One of the *business* assassins was control. When a company embarks on a multiyear journey with a big software vendor it cedes significant—if not total—control to that vendor and the business processes embedded in the

ACM Journal of Data and Information Quality



Providing Research and Tools
for Better Data

ACM JDIQ is a multi-disciplinary journal that attracts papers ranging from theoretical research to algorithmic solutions to empirical research to experiential evaluations. Its mission is to publish high impact articles contributing to the field of data and information quality (IQ).



For further information
or to submit your
manuscript,
visit jdiq.acm.org

code. For example, ERP modules were originally designed to eliminate process chaos. Remember when there were no intra- or intercompany (or industry) standardized processes? Remember when software applications never integrated? Remember when 1970s and 1980s “legacy” software was a barrier to scalability, not to mention how expensive it was to customize and maintain? ERP vendors came to the rescue by controlling the mess that homegrown applications created. But one of the side effects was the loss of process control to the vendors who defined supply chain management, financial reporting, and other business processes for the companies (and industries) they de facto managed.

While tightly bundled standardized software made some sense back in the day, it makes little or no sense in the era of *digital transformation* where disruptive business processes and business models are seen as necessary paths to competitiveness: *disruption and standardized big software are not birds of a feather*. Of course, in 1995 would have seemed heretical. Companies were desperate to end the chaos of uncoordinated business processes and rules. Standardized processes incarnated in software were the vitamin pills everyone needed. But in retrospect it is not clear that everyone understood exactly what they were consuming. When business models moved slowly in the 20th century, slow-and-steady worked, but when whole new “disruptive” business models began to appear in the 21st century (fueled by new and more powerful digital technologies), slow-and-steady became a clear threat to competitiveness.

Governance also killed big software. Big software projects that are “standardized”—that is, *required*—by corporate technology groups also usually failed, not because they did not work as advertised (which they often did not) but because of the governance that *forced* a one-size-fits-all approach to technology use. Huge off-the-shelf software packages—like ERP, CRM and DBMS packages—or even large custom in-house developed applications mandated by corporate IT—usually failed under the weight of their own governance which, to make matters worse, often resulted in increased “Shadow IT” spending.^{1,2}

The *cloud* also killed big software. Years ago, companies would implement huge software systems in their own data centers. Armies of programmers would work with armies and navies of (happy) consultants to bring big systems to life. Some years later the software might “launch” with a “switch” that—according to the data—usually failed (at least the first time). So the armies and navies would go back to work to get it right (until they got it right). Implementation cost was also a killer. \$10M often turned into \$50M, which often turned into \$250M and sometimes into billions: the Standish Group reports that big technology projects run anywhere from 40%–50% over budget—and deliver less than 50%–60% of the promised ROI.¹⁴ Cloud delivery changed all that: it is now possible to access an enterprise application directly from the cloud from any number of providers.

While implementation pain was avoided through cloud delivery, process control was still ceded to the big software vendors who owned the embedded business processes in the cloud-delivered software (while some of the control went to the cloud provider who deployed the systems on behalf of their clients). While it was almost always cheaper (by total cost of ownership [TCO] metrics) to move from on-premise big software applications to cloud hosted applications, companies were still denied access to the transformational and disruptive playing fields.^{4,18,a}

a TCO debates around on-premise-versus-cloud continue. There are all sorts of ways to compare costs across services, and many of the results will vary depending on individual services such as SaaS, IaaS, PaaS, and other cloud-based services. But a full comparison should include variables like agility, governance and long-term costs connected with training, testing, upgrades and security, among others. If a company is looking to get out of the technology business by moving its operational and strategic technology to the cloud, it will find ways to justify the cost model. If there is a bias toward keeping everything in-house then favorable on-premise cost models can be developed. The larger question is around core competency: Does the company want to be in the technology acquisition, deployment and support business—or not? The cloud offers opportunities to reassess core competencies and enables rational for various cost models, though ideally the models are based on empirical evidence.

Finally, some of the assassins were (sometimes unknowingly) *architects*. The overwhelming technical complexity and inflexibility of huge, standardized software systems also explain the death of big software. Enormous whole-company projects were often beyond the capabilities of even the most experienced project and program managers—especially when there is never 100% consensus about the need for a total enterprise project in the first place. High-level functional and non-functional requirements were nearly impossible to comprehensively define and validate; detailed requirements were even more elusive.

But perhaps the real architectural assassin was monolithic software design. Many of the big software architectures of the 20th century were conceived as integrated functional wholes versus decoupled services. Over time, monolithic architectures became impossible to cost-effectively modify or maintain and—much more importantly—became obstacles to business process change. The trend toward microservice-based architectures represents an exciting replacement to monolithic architectures (see below).

The Rise of Small, Cloudy Software

There are also small software cloud-based alternatives that scale, integrate, and share process control through customization tools deliberately built into smaller, more manageable platforms. Companies can find lots of incredibly inexpensive alternatives, from vendors like Zoho and Zendesk, among many others.^b

While “small” software packages also embed business rules and processes, they are built in smaller, more integrate-able pieces, which provides much more flexibility to clients who want to mix-and-match (existing and new) functionality.

The major driver of software change is continuous digital trans-

^b Zoho (www.zoho.com), Zendesk (www.zendesk.com). Also see: <https://www.getapp.com/customer-management-software/a/zoho-crm/alternatives/> and <https://www.crowdviews.com/zoho-crm/alternatives>

Software architectures must be blank canvasses capable of yielding tiny pictures or large masterpieces.

formation. Big standardized software systems conceived in the 20th century were not designed to adapt or self-destruct the moment a company or industry pivots.

Another way of thinking about all this is the relationship between micro and macro (or monolithic) services. *Big software begins with macroservices in monolithic architectures.*^{3,5} Or we could just think about all this as small versus large programming.⁸

Architectural assassins argue that monolithic architectures are stiff, inflexible, and unyielding. They are also difficult and expensive to maintain primarily because functionality is so interconnected and interdependent. They also argue that monolithic architectures should be replaced by microservice-based architectures.^{16,17} According to Annenko,³ “the concept is rather easy, it’s about building an application consisting of many small services that can be independently deployed and maintained, don’t have any dependencies but rather communicate with each other through lightweight mechanisms and lack a centralized infrastructure. It is even possible to write these small (micro-) services each in its own language.” Why microservice-based architectures? Annenko continues: “their benefits are undoubted, too: they easily allow for continuous deployment, and certain parts of an application can be changed, debugged or even replaced quickly and without affecting the rest. With microservices, you absolutely cannot break an ap-

plication: if something goes wrong, it will go wrong only within its own microspace, while the rest of the application will continue working as before.”

Was there any doubt that these architectural assassins would hit their target?

All of that said, SOA architecture dreams continue to develop.⁹ The big data world, for example, has already defined an open source architecture that is fast, flexible, cost-effective—and always changing.¹⁵ The tools enable low latency and real-time processing through Spark and Flink, among other open source tools. The details are specified in tools like Lambda, Kappa, and SummingBird. MapReduce moved us from parallel processing, and file systems have evolved from Google File Systems to Hadoop. Building on Hadoop, Spark and Flink provide real-time runtime environments. Even data streaming has been addressed with tools like Storm and Spark Streaming. But while SOA complements microservice-based architecture, they are different.⁷ SOA is not the threat to monolithic big software that microservice-based architecture is; in fact, SOA often behaves like a big software vitamin supplement. Said differently, SOA is not a replacement for monolithic big software and is therefore not a big software assassin.^c But candidly, SOA-based integration and interoperability have proved illusive in spite of continued promises and a growing library of open source application programming interfaces (APIs) and Web services. SOA is still more of a dream than an answer for continuous digital transformation. It might, in fact, be the wrong answer.

In addition, cloud delivery is becoming increasingly flexible. Container technology offered by companies like Docker offers freedom to companies who may need to pivot away from

^c Clark⁷ describes the differences simply: “microservices architecture is an alternative approach to structuring applications. An application is broken into smaller, completely independent components, enabling them to have greater agility, scalability, and availability. SOA exposes the functions of applications as more readily accessible service interfaces, making it easier to use their data and logic in the next generation of applications.”

ACM Transactions on Accessible Computing



This quarterly publication is a quarterly journal that publishes refereed articles addressing issues of computing as it impacts the lives of people with disabilities. The journal will be of particular interest to SIGACCESS members and delegates to its affiliated conference (i.e., ASSETS), as well as other international accessibility conferences.

www.acm.org/taccess
www.acm.org/subscribe



Association for
Computing Machinery

The entire world of traditional big software design, development, and deployment, and support is dead.

their cloud providers to another provider for any number of reasons. Containers enable clients to retain control over their applications just as emerging application architectures enable them to retain control over their software-enabled business processes.¹⁹ This means that dependencies are shrinking. So the combination of microservice-based architectures and container technology may be the response to monolithic applications.

Will the big software vendors respond? Yes.

They will milk the current big enterprise revenue streams for as long as they can and then systematically make their offerings to look more and more like their small software competitors. Many of them, like SAP and Oracle, have already by necessity begun this process through small business and mid-market cloud offerings that are much cheaper than the gold-plated goliaths they sold for years. They began to cannibalize their own products because they too know that the days of big software are numbered. But they have not fundamentally rearchitected their applications. They have shrunken them.

The Death and Resurrection of Software

The entire world of big software design, development, deployment and support is dead. Customers know it, big software vendors know it and next generation software architects know it. The implications are far-reaching and likely permanent. Business requirements, governance, cloud delivery and architecture are the assassins of old

“big” software and the liberators of new “small” software. In 20 years very few of us will recognize the software architectures of the 20th century or how software in the cloud enables ever-changing business requirements. ■

References

- Andriole, S. Who owns IT? *Commun. ACM* 58, 8 (Aug. 2015).
- Andriole, S., Cox, T. and Khin, K. *Technology Adoption & Digital Transformation*. CRC Press, 2017.
- Annenko, O. Breaking down the monolithic: Microservices vs. self-contained systems. DZone, June 2016; <http://bit.ly/2dEfFBG>
- Boisvert, G. Cost of Server Ownership: On-Premise Vs. IaaS. SherWeb, Sept. 2015; <http://bit.ly/2z3Sg9l>
- Brown, S. What is agile software architecture, *Coding the Architecture*, 2013; http://www.codingthearchitecture.com/2013/09/03/what_is_agile_software_architecture.html
- Charette, R.N. U.S. Air Force blows \$1 billion on failed ERP project. *IEEE Spectrum*, Nov. 2012; <http://bit.ly/2zim1E1>
- Clark, T. Microservices, SOA, and APIs: Friends or enemies?: A comparison of key integration and application architecture concepts for an evolving enterprise. IBM DeveloperWorks, Jan. 2016; <https://ibm.co/2zhsMWR>
- DeRemer, F. and Kron, H.K. Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering*, 2 (June 1976); <http://bit.ly/2yLzVM>
- Erl, T. et al. *Next Generation SOA: A Concise Introduction to Service Technology & Service-Oriented*. Prentice Hall, 2015.
- Gorton, I. Cyberinfrastructures: Bridging the divide between scientific research and software engineering. *Computer* 47, 8 (Aug. 2014); 48, 55; <http://bit.ly/2yWVjKf>
- Kimberling, E. Key Findings from the 2015 Report. Panorama Consulting, Apr. 2015; <http://bit.ly/2hpGwWo>
- Lee, J. 9 VERY scary ERP and ERP system implementation statistics. ERP/VAR, Oct. 2014; <http://bit.ly/2yukxJf>
- Leibowitz, J. IT project failures: What management can learn. *IEEE IT Professional* (Apr. 2016); <http://bit.ly/2ynZnHF>
- Lynch, J. The Chaos Report. The Standish Group, 2015; <http://bit.ly/2zScMqv>
- Madan, A. 100 open source big data architecture papers for data professionals. LinkedIn, (June 2015); <http://bit.ly/1UEZdRt>
- McLarty, M. Microservice architecture is agile software architecture. *Infoworld*, May 2016; <http://bit.ly/24hvrnD>
- Proctor, S. From monolith to microservices: Big rewards from small software architecture. *IT World Canada*, (Aug. 2016); <http://bit.ly/2iglbgk>
- Tomkins, B. SaaS solutions 77% cheaper than on-premises. *Information Week*, (May 2010); <http://ubm.io/2z4wAd9>
- Townsend, K. Containers: The pros and the cons of these VM alternatives. *TechRepublic*, Feb. 2015; <http://tek.io/2nfzjzv>
- Wailgum, T. 10 famous ERP disasters, dustups and disappointments. *CIO Magazine* (Mar. 2009); <http://bit.ly/2zv2mXK>

Stephen J. Andriole (steve@andriole.com) is the Thomas G. Labrecque Professor of Business at the Villanova School of Business at Villanova University where he teaches and conducts research in emerging technologies, requirements modeling and business technology strategy. His most recent book is *Ready Technology: Fast Tracking Emerging Business Technologies* (CRC Press, 2014).

The author thanks the reviewers who significantly improved the article. With their help, the “death of big software” message was clarified especially regarding the discussion of microservice-based architectures.

Copyright held by author.

Viewpoint

Lousy Advice to the Lovelorn

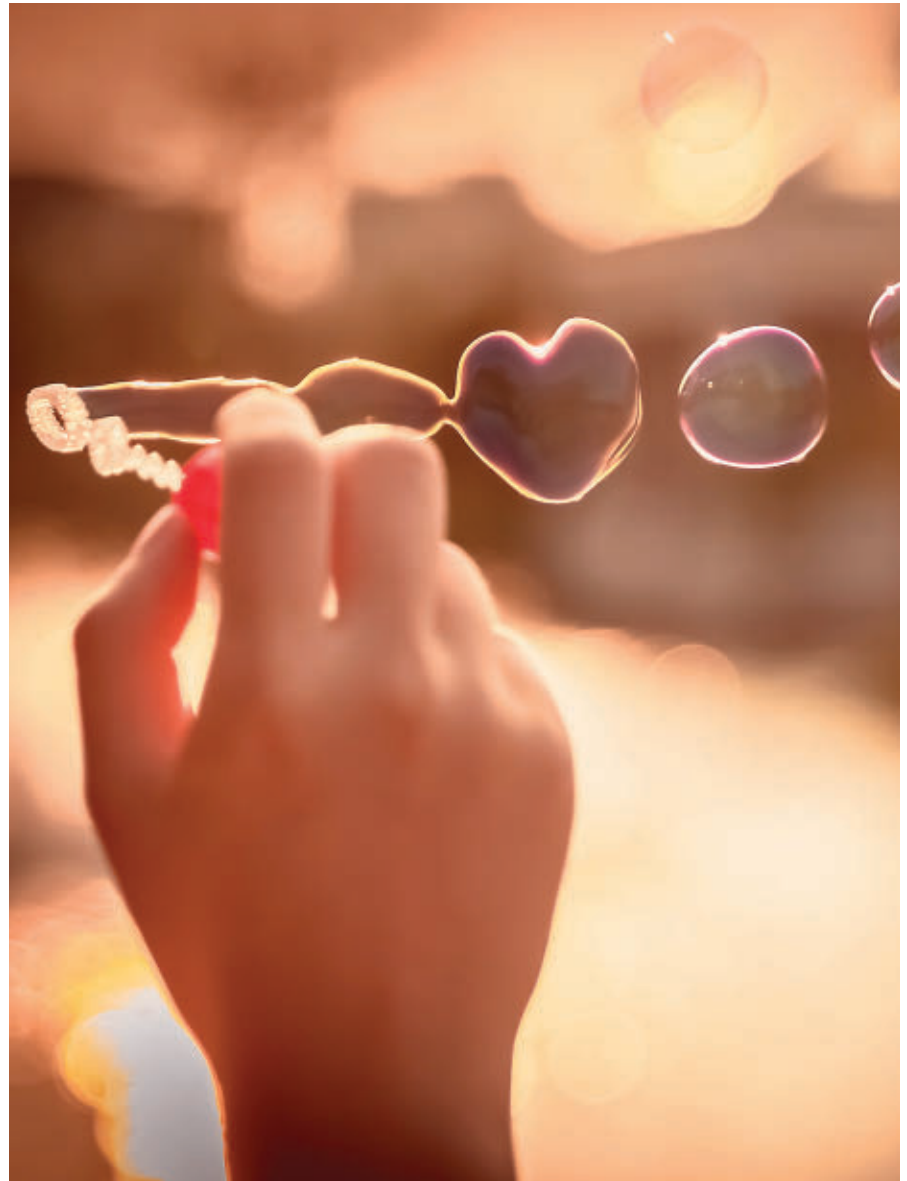
The 37% rule is rarely applicable in real-world situations. It is certainly entirely wrong-headed as advice for getting married.

PEOPLE SEEKING ADVICE ON getting married have recently been directed to a decidedly unromantic algorithm. The algorithm has two stages. Let N be the maximum number of people you expect to be able to date before you give up. In the first stage, you date and dump $N/e \approx .37N$ people to get a sense of the overall quality of the field. In the second stage, you continue down the list, and you marry the first person that is better than everyone you met in the first stage. (If you reach the end of the list, and the last person is not the best, then the algorithm is indifferent; you can marry them or not.) There is a theorem that supposedly states that following this strategy maximizes the likelihood of marrying the best possible partner of those on your original list.

This algorithm is the subject of the first chapter of *Algorithms to Live By*, by Brian Christian and Tom Griffiths,¹ and has been repeated in recent articles in *The Washington Post*,⁶ *Business Insider*,⁷ *Slate*,⁵ and *NPR*.⁴ The headline in *Business Insider* makes the advice even simpler: 26 is the perfect age to get married.

If this advice seems dubious, your intuitions are working well. Taken literally, the advice is crazy, because the assumptions of the underlying theorem bear no relation to the real world of dating and marriage.

I tell a tale to illustrate the problems with the algorithm. (A version of this tale in verse can be found at Davis.²) Strehphon lives in Arcadia with 100 eligible



ACM Transactions on Social Computing



ACM TSC seeks to publish work that covers the full spectrum of social computing including theoretical, empirical, systems, and design research contributions. TSC welcomes research employing a wide range of methods to advance the tools, techniques, understanding, and practice of social computing, particularly research that designs, implements or studies systems that mediate social interactions among users, or that develops theory or techniques for application in those systems.



For further information
or to submit your
manuscript,
visit tsc.acm.org

bachelorettes. He is happy to date them all, so $N = 100$. Therefore, following the algorithm, his plan is first to date 37 of them, and then to marry the next one who is better than any of those 37.

However, the course of following algorithms never did run smooth. Date 17 is Chloe, who is amazing. Strephon is completely smitten. However, a theorem is a theorem, and Strephon can easily calculate that the probability is 0.83 that someone he has not met yet is even better than Chloe. Goodbye Chloe.

The next 34 dates go by with no magic. Date 52 is Phoebe. Phoebe is even more amazing than Chloe. Patting himself on the back for his wisdom and patience in passing up Chloe, Strephon proposes to Phoebe. For Phoebe, Strephon is date #40 out of 75, so she is in her second stage^a and she really likes him. Sadly, though, she liked Colin, her #15, even better, so no dice.

Forty-five more women come and go. Date 98 is Daphne. Daphne is as amazing as Chloe except that she listens to the controversial avant-garde composer Karlheinz Stockhausen, but Strephon can live with that. The probability that one of the remaining two women is better than Daphne is only about 0.06. However, the goal of the algorithm is to marry the best partner, and Daphne is definitely not the best partner, since she is inferior to Chloe. Strephon knows better than to contravene an algorithm endorsed by a professor at Berkeley and published in the *Washington Post*. Goodbye Daphne.

What the algorithm prescribes in the case where Daphne is better than Chloe but worse than Phoebe is not clearly specified; it depends on whether the goal is to marry the best partner of all those on the list, or to marry the best partner who would have accepted you.

You might suppose that Strephon could try calling Chloe and begging for forgiveness; but the theorem that Strephon is relying on assumes that is impossible. By assumption, Chloe has already married someone else.

Enough of the melodrama; what is the math? The theorem is as follows: Suppose you will be presented with a sequence of options, from which you are

to select one. At each stage, you must choose the current object or you can pass on it, but either way, your decision is irrevocable. Suppose further that:

- The only pertinent information you get about the candidates is ordinal. That is, you can judge how the new candidate compares to the earlier ones, but you cannot assign a meaningful cardinal score.

- Your goal is to maximize the probability of getting the best of the N candidates; you are indifferent between all other outcomes.

In that case, the preceding algorithm—pass on the first N/e , then choose the next one that is better than all those—is provably optimal. Following that algorithm, the probability is about $1/e$ that you will get the best partner.

The problem and its solution were first published in Martin Gardner's "Mathematical Games" column.³ Chapter 1 of Christian and Griffiths¹ and its 15 pages of notes contain a thorough discussion of the problem's history, variants, and applications; see also the Wikipedia entry, Secretary Problem.⁸

Given the extreme restrictions, it is easy to see that the optimal rule to follow must have the form, "Pass on the first $f(N)$, then choose the next one that is better" for some function $f(N)$. Obviously, a candidate that is not better than all those seen so far cannot be chosen, since it cannot be the best. Moreover, all the information that you have in choosing is the sequence of ordinals; and it is easy to see that the order in which the ones you have passed on carries no useful information.

The assumptions of the theorem are almost never even approximately satisfied in real-world situations—and are especially far-fetched in regard to dating.

^a There is a dating service in Arcadia that matches first-stagers with first-stagers and second-stagers with second-stagers.

For example, suppose that for $N = 100$ someone were to propose an alternative rule: “Pass on the first 37, then wait until someone—call her Alice—is better than those 37, then continue to wait until someone else—Betty—is better than Alice.” Suppose that Strephon executes this, meets Alice at 40, and meets Betty at 51. Then all the useful information that he actually has is that Betty is better than 50 women out of 100. Consider an alternative order in which Betty is still 50, but Alice is now 17. Then Betty is now the first person he has met better than the first 37, so by the rule, he should pass on Betty. But there is no significant difference between the first and second situation in what Strephon knows, since the ordering was random.

The fact that $f(N)$ converges to N/e is, to my mind, cute, but not very profound. The proof is an exercise in combinatorics, with nothing particularly insightful.

It seems to me the assumptions of the theorem are almost never even approximately satisfied in real-world situations—and are especially far-fetched in regard to dating. There are many problems. I will first mention a few that seem to me comparatively superficial and open to technical solutions. The assumption that you cannot go back—that Strephon cannot go back to Chloe—is often false, but it is reasonable in cases like real-estate hunting or searching for a parking space where there is a lot of competition for desirable resources. It is arguably reasonable as regards dating. The assumption that you know the value of N in advance is often dubious, but certainly there are cases where N can be reasonably estimated. The analysis ignores transactional costs; dating takes time and effort.

Then there are more fundamental problems. The assumption that seems most far-fetched is the “best or nothing” assumption: you want to maximize the probability of getting the best option and are completely indifferent between the other outcomes.

I cannot think of any choice problem where this is a reasonable assumption. This becomes clear if the assumption is cast in terms of maximum expected utility theory, the standard normative theory of choice. This assumption corresponds to assigning a utility of 1 if the best item is chosen

It is not good to promote the view of romantic partners as commodities that are inspected and chosen rather than people whom you woo and who woo you.

and of 0 if it is not. But the agent never knows whether the item he has chosen is actually the best, and therefore does not know his own utility. For instance, suppose that Strephon decides to ignore the algorithm and marry Chloe even though she is only date number 17. Then his utility is 1 if she actually is the best partner for him and 0 if there is someone better, despite the fact that he has no idea which is the case and that his state of nuptial bliss is presumably unaffected. I cannot find any axiom that prohibits this kind of utility function, but it seems counter to the general spirit of what is usually meant by a utility.^b

The assumption that there is only ordinal information is also problematic. This is critical in Strephon’s decision to reject Chloe; it is assumed that there is no difference between her being slightly better than the others he has seen and her being enormously better. In most real situations, once you have seen a reasonable number of options, you have some idea of the ordinary range and you can spot an astonishing outlier.

These objections apply in all cases of choice. For courtship specifically, there is another serious problem. Chloe, Phoebe, and Daphne are not parking spots, waiting there to be chosen; they

^b Richard Cole points out that the algorithm has the property that, regardless of the distribution of utilities over items, which is generally poorly known and even poorly defined, the expected value of the utility of the outcome over random orderings is at least $1/e$ of the optimal choice, assuming all utilities are non-negative.

are not even stochastic processes that randomly accept or reject marriage proposals. At least in our society, they are players engaged in much the same game as Strephon. Suppose that Phoebe’s and Strephon’s preferences are uncorrelated; that is, Phoebe’s preferences are independent of the fact that Strephon proposed to her. Then, even given that Phoebe is in stage 2, the probability that she prefers Strephon to all the men she met in stage 1 is only about e/N . Of course Strephon may have another chance to propose; but the average number of stage 2 candidates who are better than all the stage 1 candidates is only about $e - 1$. There will be a lot of unmarried people in Arcadia.

Probably preferences between potential partners are substantially correlated, which makes things better; but they are also probably correlated across rivals, which makes things worse. In any case, the analysis in this situation would be completely different (and very difficult).

Lovers are proverbially unwise, but I think that most would have the good sense to ditch the algorithm when its advice is obviously idiotic. So I am not actually very worried about Strephon and the rest. However, it is not good to promote the view of romantic partners as commodities that are inspected and chosen rather than people whom you woo and who woo you. Moreover, publishing senseless advice as the unquestionable dictate of mathematics strengthens the general view that mathematics and science are just theories that have nothing to do with the real world. ■

References

1. Christian, B. and Griffiths, T. *Algorithms to Live By: The Computer Science of Human Decisions*. Henry Holt, 2016.
2. Davis, E. The 37% Rule. 2017; <http://cs.nyu.edu/faculty/davise/Verses/ThirtySeven.html>
3. Gardner, M. *New Mathematical Diversions from Scientific American*. Simon and Schuster, 1966.
4. Krulwich, R. How to marry the right girl: A mathematical solution. National Public Radio, May 15, 2014.
5. Parker, M. The secretary problem: An algorithm for deciding who to marry and other tough choices. *State* (Dec. 17, 2014).
6. Swanson, A. When to stop dating and settle down, according to math. *The Washington Post* (Feb. 16, 2016).
7. Weller, C. A mathematical theory says the perfect age to get married is 26—here’s why. *Business Insider* (Nov. 25, 2016).
8. Wikipedia. Secretary problem; http://en.wikipedia.org/wiki/Secretary_problem

Ernest Davis (davise@cs.nyu.edu) is Professor of Computer Science at New York University.

Copyright held by author.

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

The concept of cryptocurrencies is built from forgotten ideas in research literature.

BY ARVIND NARAYANAN AND JEREMY CLARK

Bitcoin's Academic Pedigree

IF YOU HAVE read about bitcoin in the press and have some familiarity with academic research in the field of cryptography, you might reasonably come away with the following impression: Several decades' worth of research on digital cash, beginning with David Chaum,^{10,12} did not lead to commercial success because it required a centralized, bank-like server controlling the system, and no banks wanted to sign on. Along came bitcoin, a radically different proposal for a decentralized cryptocurrency that did not need the banks, and digital cash finally succeeded. Its inventor, the mysterious Satoshi Nakamoto, was an academic outsider, and bitcoin bears no resemblance to earlier academic proposals.

This article challenges that view by showing nearly all of the technical components of bitcoin originated in the academic literature of the 1980s and 1990s (see Figure 1). This is not to diminish Nakamoto's achievement but to point out he stood on the shoulders of giants. Indeed, by tracing the origins of the ideas in bitcoin, we can zero in on Nakamoto's true leap of insight—the specific, complex way in which the underlying components are put together. This helps explain why bitcoin took so long to be invented. Readers already familiar with how bitcoin works may gain a deeper understanding from this historical presentation. (For an introduction, see *Bitcoin and Cryptocurrency Technologies*.³⁶) Bitcoin's intellectual history also serves as a case study demonstrating the relationships among academia, outside researchers, and practitioners, and offers lessons on how these groups can benefit from one another.

The Ledger

If you have a secure ledger, the process to leverage it into a digital payment system is straightforward. For example, if Alice sends Bob \$100 by PayPal, then PayPal debits \$100 from Alice's account and credits \$100 to Bob's account. This is also roughly what happens in traditional banking, although the absence of a single ledger shared between banks complicates things.

This idea of a ledger is the starting point for understanding bitcoin. It is a place to record all transactions that happen in the system, and it is open to and trusted by all system participants. Bitcoin converts this system for recording payments into a currency. Whereas in banking, an account balance represents cash that can be demanded from the bank, what does a unit of bitcoin represent? For now, assume that what is being transacted holds value inherently.

How can you build a ledger for use in an environment like the Internet where participants may not trust each other? Let's start with the easy part: the choice of data structure. There are a



few desirable properties. The ledger should be immutable or, more precisely, *append only*: you should be able to add new transactions but not remove, modify, or reorder existing ones. There should also be a way to obtain a succinct *cryptographic digest* of the state of the ledger at any time. A digest is a short string that makes it possible to avoid storing the entire ledger, knowing that if the ledger were tampered with in any way, the resulting digest would change, and thus the tampering would be detected. The reason for these properties is that unlike a regular data structure that is stored on a single machine, the ledger is a *global* data structure collectively maintained by a mutually untrusting set of partici-

pants. This contrasts with another approach to decentralizing digital ledgers,^{7,13,21} in which many participants maintain local ledgers and it is up to the user querying this set of ledgers to resolve any conflicts.

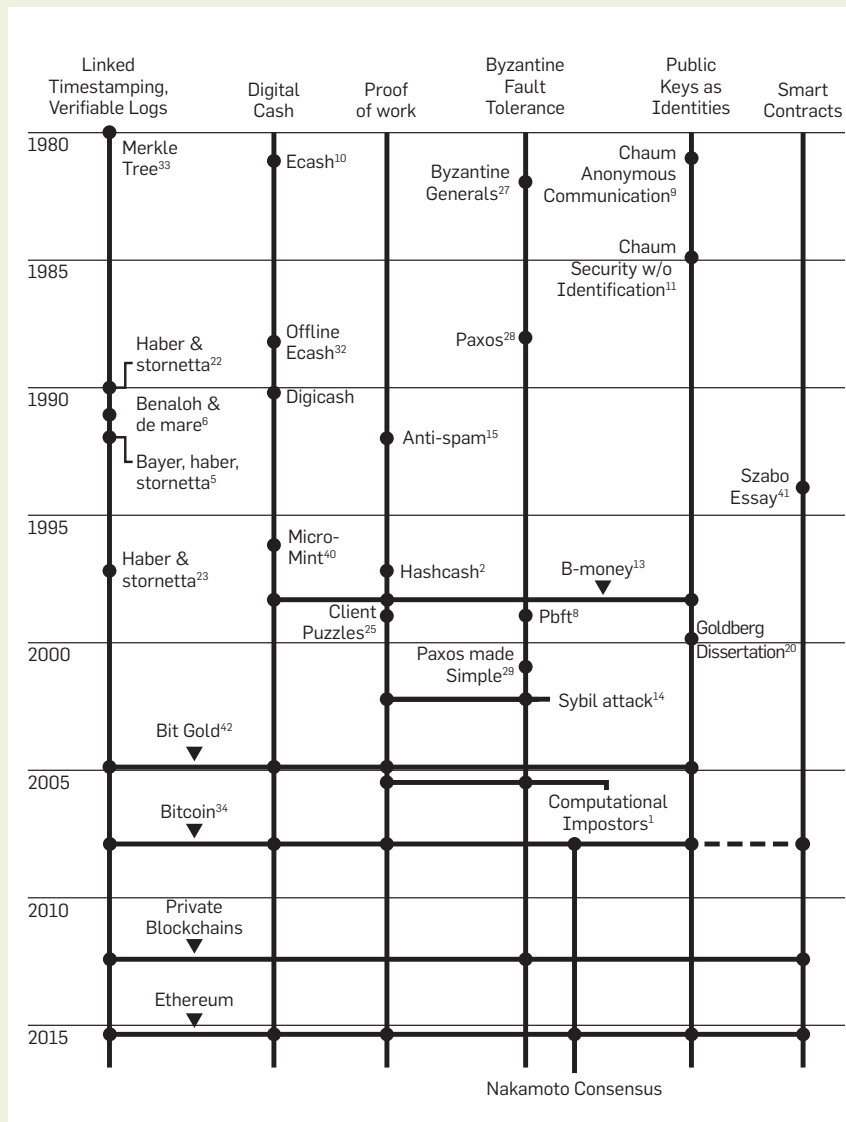
Linked timestamping. Bitcoin’s ledger data structure is borrowed, with minimal modifications, from a series of papers by Stuart Haber and Scott Stornetta written between 1990 and 1997 (their 1991 paper had another co-author, Dave Bayer).^{5,22,23} We know this because Nakamoto says so in his bitcoin white paper.³⁴ Haber and Stornetta’s work addressed the problem of document timestamping—they aimed to build a “digital notary” service. For patents, business contracts, and other documents, one may

want to establish that the document was created at a certain point in time, and no later. Their notion of document is quite general and could be any type of data. They do mention, in passing, financial transactions as a potential application, but it was not their focus.

In a simplified version of Haber and Stornetta’s proposal, documents are constantly being created and broadcast. The creator of each document asserts a time of creation and signs the document, its timestamp, and the previously broadcast document. This previous document has signed its own predecessor, so the documents form a long chain with pointers backwards in time. An outside user cannot alter a timestamped message since it is signed by the creator, and the creator cannot alter the message without also altering the entire chain of messages that follows. Thus, if you are given a single item in the chain by a trusted source (for example, another user or a specialized timestamping service), the entire chain up to that point is locked in, immutable, and temporally ordered. Further, if you assume the system rejects documents with incorrect creation times, you can be reasonably assured that documents are at least as old as they claim to be. At any rate, bitcoin borrows only the data structure from Haber and Stornetta’s work and reengineers its security properties with the addition of the proof-of-work scheme described later in this article.

In their follow-up papers, Haber and Stornetta introduced other ideas that make this data structure more effective and efficient (some of which were hinted at in their first paper). First, links between documents can be created using hashes rather than signatures; hashes are simpler and faster to compute. Such links are called hash pointers. Second, instead of threading documents individually—which might be inefficient if many documents are created at approximately the same time—they can be grouped into batches or blocks, with documents in each block having essentially the same timestamp. Third, within each block, documents can be linked together with a binary tree of hash pointers, called a Merkle tree, rather than a linear chain. Incidentally, Josh Benaloh and Michael

Figure 1. Chronology of key ideas found in bitcoin.



de Mare independently introduced all three of these ideas in 1991,⁶ soon after Haber and Stornetta's first paper.

Merkle trees. Bitcoin uses essentially the data structure in Haber and Stornetta's 1991 and 1997 papers, shown in simplified form in Figure 2 (Nakamoto was presumably unaware of Benaloh and de Mare's work). Of course, in bitcoin, transactions take the place of documents. In each block's Merkle tree, the leaf nodes are transactions, and each internal node essentially consists of two pointers. This data structure has two important properties. First, the hash of the latest block acts as a digest. A change to any of the transactions (leaf nodes) will necessitate changes propagating all the way to the root of the block, and the roots of all following blocks. Thus, if you know the latest hash, you can download the rest of the ledger from an untrusted source and verify that it has not changed. A similar argument establishes another important property of the data structure—that is, someone can efficiently prove to you that a particular transaction is included in the ledger. This user would have to send you only a small number of nodes in that transaction's block (this is the point of the Merkle tree), as well as a small amount of information for every following block. The ability to efficiently prove inclusion of transactions is highly desirable for performance and scalability.

Merkle trees, by the way, are named for Ralph Merkle, a pioneer of asymmetric cryptography who proposed the idea in his 1980 paper.³³ His intended application was to produce a digest for a public directory of digital certificates. When a website, for example, presents you with a certificate, it could also present a short proof that the certificate appears in the global directory. You could efficiently verify the proof as long as you know the root hash of the Merkle tree of the certificates in the directory. This idea is ancient by cryptographic standards, but its power has been appreciated only of late. It is at the core of the recently implemented Certificate Transparency system.³⁰ A 2015 paper proposes CONIKS, which applies the idea to directories of public keys for end-to-end encrypted emails.³² Efficient verification of parts of the global

state is one of the key functionalities provided by the ledger in Ethereum, a new cryptocurrency.

Bitcoin may be the most well-known real-world instantiation of Haber and Stornetta's data structures, but it is not the first. At least two companies—Surety starting in the mid-1990s and Guardtime starting in 2007—offer document timestamping services. An interesting twist present in both of these services is an idea mentioned by Bayer, Haber, and Stornetta,⁵ which is to publish Merkle roots periodically in a newspaper by taking out an ad. Figure 3 shows a Merkle root published by Guardtime.

Byzantine fault tolerance. Of course, the requirements for an Internet currency without a central authority are more stringent. A distributed ledger will inevitably have forks, which means that some nodes will think block A is the latest block, while other nodes will think it is block B. This could be be-

cause of an adversary trying to disrupt the ledger's operation or simply because of network latency, resulting in blocks occasionally being generated near-simultaneously by different nodes unaware of each other's blocks. Linked timestamping alone is not enough to resolve forks, as was shown by Mike Just in 1998.²⁶

A different research field, fault-tolerant distributed computing, has studied this problem, where it goes by different names, including *state replication*. A solution to this problem is one that enables a set of nodes to apply the same state transitions in the same order—typically, the precise order does not matter, only that all nodes are consistent. For a digital currency, the state to be replicated is the set of balances, and transactions are state transitions. Early solutions, including Paxos, proposed by Turing Award winner Leslie Lamport in 1989,^{28,29} consider state replication

Figure 2. The ledger data structure in linked timestamping.

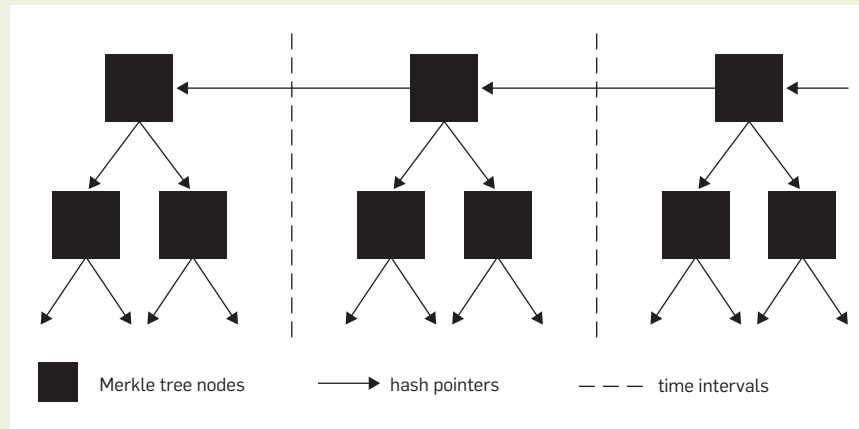
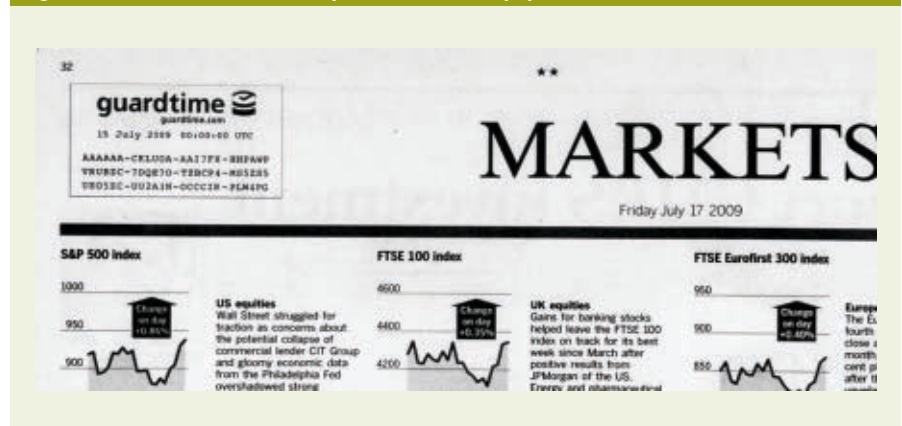


Figure 3. Guardtime Merkle root published in newspaper.




when communication channels are unreliable and when a minority of nodes may exhibit certain “realistic” faults, such as going offline forever or rebooting and sending outdated messages from when it first went offline. A prolific literature followed with more adverse settings and efficiency trade-offs.


A related line of work studied the situation where the network is mostly reliable (messages are delivered with bounded delay), but where the definition of “fault” was expanded to handle *any* deviation from the protocol. Such Byzantine faults include both naturally occurring faults as well as maliciously crafted behaviors. They were first studied in a paper also by Lamport, cowritten with Robert Shostak and Marshall Pease, as early as 1982.²⁷ Much later, in 1999, a landmark paper by Miguel Castro and Barbara Liskov introduced practical Byzantine fault tolerance (PBFT), which accommodated both Byzantine faults and an unreliable network.⁸ Compared with linked time-stamping, the fault-tolerance literature is enormous and includes hundreds of variants and optimizations of Paxos, PBFT, and other seminal protocols.

In his original white paper, Nakamoto does not cite this literature or use its language. He uses some concepts, referring to his protocol as a consensus mechanism and considering faults both in the form of attackers, as well as nodes joining and leaving the network. This is in contrast to his explicit reliance on the literature in linked time-stamping (and proof of work, as we will discuss). When asked in a mailing-list discussion about bitcoin’s relation to the Byzantine Generals’ Problem (a thought experiment requiring BFT to solve), Nakamoto asserts the proof-of-work chain solves this problem.³⁵

In the following years, other academics have studied Nakamoto consensus from the perspective of distributed systems. This is still a work in progress. Some show that bitcoin’s properties are quite weak,⁴⁵ while others argue that the BFT perspective does not do justice to bitcoin’s consistency properties.⁴¹ Another approach is to define variants of well-studied properties and prove that bitcoin satisfies them.¹⁹ Recently these definitions were substantially



Virtually all fault-tolerant systems assume that a majority or supermajority of nodes in the system are both honest and reliable.



sharpened to provide a more standard consistency definition that holds under more realistic assumptions about message delivery.³⁷ All of this work, however, makes assumptions about “honest,” that is, protocol-compliant, behavior among a subset of participants, whereas Nakamoto suggests that honest behavior need not be blindly assumed, because it is incentivized. A richer analysis of Nakamoto consensus accounting for the role of incentives does not fit cleanly into past models of fault-tolerant systems.

Proof Of Work

Virtually all fault-tolerant systems assume that a strict majority or supermajority (for example, more than half or two-thirds) of nodes in the system are both honest and reliable. In an open peer-to-peer network, there is no registration of nodes, and they freely join and leave. Thus an adversary can create enough *Sybil*s, or sockpuppet nodes, to overcome the consensus guarantees of the system. The Sybil attack was formalized in 2002 by John Douceur,¹⁴ who turned to a cryptographic construction called *proof of work* to mitigate it.

The origins. To understand proof of work, let’s turn to its origins. The first proposal that would be called proof of work today was created in 1992 by Cynthia Dwork and Moni Naor.¹⁵ Their goal was to deter spam. Note that spam, Sybil attacks, and denial of service are all roughly similar problems in which the adversary amplifies its influence in the network compared to regular users; proof of work is applicable as a defense against all three. In Dwork and Naor’s design, email recipients would process only those email messages that were accompanied by proof that the sender had performed a moderate amount of computational work—hence, “proof of work.” Computing the proof would take perhaps a few seconds on a regular computer. Thus, it would pose no difficulty for regular users, but a spammer wishing to send a million email messages would require several weeks, using equivalent hardware.

Note that the proof-of-work *instance* (also called a puzzle) must be specific to the email, as well as to the recipient. Otherwise, a spammer

would be able to send multiple messages to the same recipient (or the same message to multiple recipients) for the cost of one message to one recipient. The second crucial property is that it should pose minimal computational burden on the recipient; puzzle solutions should be trivial to verify, regardless of how difficult they are to compute. Additionally, Dwork and Naor considered functions with a *trapdoor*, a secret known to a central authority that would allow the authority to solve the puzzles without doing the work. One possible application of a trapdoor would be for the authority to approve posting to mailing lists without incurring a cost. Dwork and Naor's proposal consisted of three candidate puzzles meeting their properties, and it kicked off a whole research field, to which we will return.

Hashcash. A very similar idea called hashcash was independently invented in 1997 by Adam Back, a postdoctoral researcher at the time who was part of the cypherpunk community. Cypherpunks were activists who opposed the power of governments and centralized institutions, and sought to create social and political change through cryptography. Back was practically oriented: he released hashcash first as software,² and five years later in 2002 released an Internet draft (a standardization document) and a paper.⁴

Hashcash is much simpler than Dwork and Naor's idea: it has no trapdoor and no central authority, and it uses only hash functions instead of digital signatures. It is based on a simple principle: a hash function behaves as a random function for some practical purposes, which means the only way to find an input that hashes to a particular output is to try various inputs until one produces the desired output. Further, the only way to find an input that hashes into an arbitrary *set* of outputs is again to try hashing different inputs one by one. So, if I challenged you to find an input whose (binary) hash value begins with 10 zeros, you would have to try numerous inputs, and you would find that each output had a $1/2^{10}$ chance of beginning with 10 zeros, which means that you would have to try on the order of 2^{10} inputs, or approximately 1,000 hash computations.

Sybil-Resistant Networks

In his paper on Sybil attacks, John Douceur proposed that all nodes participating in a BFT protocol be required to solve hashcash puzzles. If a node were masquerading as N nodes, it would be unable to solve N puzzles in time, and the fake identities would be purged. Karma, an early peer-to-peer digital cash system, uses a hashcash-like puzzle to rate-limit nodes joining the Karma network and receiving credits for file sharing.⁴⁴ A malicious node, however, could still obtain a moderate advantage over an honest node that claimed only a single identity. A follow-up paper in 2005¹ suggested honest nodes should instead mimic the behavior of malicious nodes and claim as many virtual identities as they computationally can afford to claim. With these virtual identities executing a BFT protocol, the assumption, "At most a fraction f of nodes are faulty" can be replaced with the assumption "The fraction of total computational power controlled by faulty nodes is at most f ." Thus, it is no longer necessary to validate identities, and open peer-to-peer networks can run a BFT protocol. Bitcoin uses exactly this idea. But Nakamoto asks: What motivates nodes to perform computationally expensive proof of work? The answer requires a further leap: digital currency.

Smart Contracts

A smart contract takes the idea of putting *data* in a secure ledger and extends it to *computation*. In other words, it is a consensus protocol for the correct execution of a publicly specified program. Users can invoke functions in these smart-contract programs, subject to any restrictions specified by the program, and the function code is executed in tandem by the miners. Users can trust the output without having to redo the computation and can write their own programs to act on the output of other programs. Smart contracts are especially powerful when combined with a cryptocurrency platform, because the programs in question can handle money—own it, transfer it, destroy it, and, in some cases, even print it.

Bitcoin implements a restrictive programming language for smart contracts. A "standard" transaction (that is, one that moves currency from one address to another) is specified as a short script in this language. Ethereum offers a more permissive and powerful language.

The idea of smart contracts was proposed by Nick Szabo in 1994⁴² and so named because he saw them as analogs of legal contracts, but with automated enforcement. (This view has been critiqued by Levy³¹ and Felten.¹⁶) Presciently, Szabo presented smart contracts as extensions of digital-cash protocols and recognized that Byzantine agreement and digital signatures (among others) could be used as building blocks. The success of cryptocurrencies has made smart contracts practical, and research on the topic has bloomed as well. For example, programming languages researchers have adapted their methods and tools to automatically discover bugs in smart contracts and to write verifiably correct ones.

Permissioned Blockchains

While this article has emphasized that private or permissioned blockchains omit most of bitcoin's innovations, this is not meant to diminish the interesting work happening in this space. A permissioned blockchain places restrictions on who can join the network, write transactions, or mine blocks. In particular, if miners are restricted to a list of trustworthy participants, the proof of work can be dropped in favor of a more traditional BFT approach. Thus, much of the research is a rebirth of BFT that asks questions such as: Can we use hash trees to simplify consensus algorithms? What if the network can fail only in certain ways?

Further, there are important considerations around identity and public-key infrastructure, access control, and confidentiality of the data stored on the blockchain. These issues largely do not arise in public blockchain settings, nor are they studied in the traditional BFT literature.

Finally, there is also the engineering work of scaling blockchains for high throughput and adapting them to various applications such as supply-chain management and financial technology.

As the name suggests, in hashcash Back viewed proof of work as a form of cash. On his webpage he positioned it as an alternative to David Chaum's DigiCash, which was a system that issued untraceable digital cash from a bank to a user.³ He even made compromises to the technical design to make it appear more cashlike. Later, Back made comments suggesting that bitcoin was a straightforward extension of hashcash. Hashcash is simply not cash, however, because it has no protection against double spending. Hashcash tokens cannot be exchanged among peers.

Meanwhile, in the academic scene, researchers found many applications for proof of work besides spam, such as preventing denial-of-service attacks,²⁵ ensuring the integrity of Web analytics,¹⁷ and rate-limiting password guessing online.³⁸ Incidentally, the term *proof of work* was coined only in 1999 in a paper by Markus Jakobsson and Ari Juels, which also includes a nice survey of the work up until that point.²⁴ It is worth noting that these researchers seem to have been unaware of hashcash but independently started to converge on hash-based proof of work, which was introduced in papers by Eran Gabber et al.¹⁸ and by Juels and Brainard.²⁵ (Many of the terms used throughout this paragraph did not become standard terminology until long after the papers in question were published.)

Proof of work and digital cash: A catch-22. You may know that proof of work did not succeed in its original application as an anti-spam measure. One possible reason is the dramatic difference in the puzzle-solving speed of different devices. That means spammers will be able to make a small investment in custom hardware to increase their spam rate by orders of magnitude. In economics, the natural response to an asymmetry in the cost of production is trade—that is, a market for proof-of-work solutions. But this presents a catch-22, because that would require a working digital currency. Indeed, the lack of such a currency is a major part of the motivation for proof of work in the first place. One crude solution to this problem is to declare puzzle solutions to *be* cash, as hashcash tries to do.

More coherent approaches to treating puzzle solutions as cash are found in two essays that preceded bitcoin, describing ideas called b-money¹³ and bit gold⁴³ respectively. These proposals offer timestamping services that sign off on the creation (through proof of work) of money, and once money is created, they sign off on transfers. If disagreement about the ledger occurs among the servers or nodes, however, there isn't a clear way to resolve it. Letting the majority decide seems to be implicit in both authors' writings, but because of the Sybil problem, these mechanisms are not very secure, unless there is a gatekeeper who controls entry into the network or Sybil resistance is itself achieved with proof of work.

Putting It All Together

Understanding all these predecessors that contain pieces of bitcoin's design leads to an appreciation of the true genius of Nakamoto's innovation. In bitcoin, for the first time, puzzle solutions don't constitute cash by themselves. Instead, they are merely used to secure the ledger. Solving proof of work is performed by specialized entities called *miners* (although Nakamoto underestimated just how specialized mining would become).

Miners are constantly in a race with each other to find the next puzzle solution; each miner solves a slightly different variant of the puzzle so that the chance of success is proportional to the fraction of global mining power that the miner controls. A miner who solves a puzzle gets to contribute the next batch, or block, of transactions to the ledger, which is based on linked timestamping. In exchange for the service of maintaining the ledger, a miner who contributes a block is rewarded with newly minted units of the currency. With high likelihood, if a miner contributes an invalid transaction or block, it will be rejected by the majority of other miners who contribute the following blocks, and this will also invalidate the block reward for the bad block. In this way, because of the monetary incentives, miners ensure each other's compliance with the protocol.

Bitcoin neatly avoids the double-spending problem plaguing proof-of-work-as-cash schemes because it es-

chews puzzle solutions themselves having value. In fact, puzzle solutions are *twice* decoupled from economic value: the amount of work required to produce a block is a floating parameter (proportional to the global mining power), and further, the number of bitcoins issued per block is not fixed either. The block reward (which is how new bitcoins are minted) is set to halve every four years (in 2017, the reward is 12.5 bitcoins/block, down from 50 bitcoins/block). Bitcoin incorporates an additional reward scheme—namely, senders of transactions paying miners for the service of including the transaction in their blocks. It is expected the market will determine transaction fees and miners' rewards.

Nakamoto's genius, then, was not any of the individual components of bitcoin, but rather the intricate way in which they fit together to breathe life into the system. The timestamping and Byzantine agreement researchers didn't hit upon the idea of incentivizing nodes to be honest, nor, until 2005, of using proof of work to do away with identities. Conversely, the authors of hashcash, b-money, and bit gold did not incorporate the idea of a consensus algorithm to prevent double spending. In bitcoin, a secure ledger is necessary to prevent double spending and thus ensure that the currency has value. A valuable currency is necessary to reward miners. In turn, strength of mining power is necessary to secure the ledger. Without it, an adversary could amass more than 50% of the global mining power and thereby be able to generate blocks faster than the rest of the network, double-spend transactions, and effectively rewrite history, overrunning the system. Thus, bitcoin is bootstrapped, with a circular dependence among these three components. Nakamoto's challenge was not just the design, but also convincing the initial community of users and miners to take a leap together into the unknown—back when a pizza cost 10,000 bitcoins and the network's mining power was less than a trillionth of what it is today.

Public keys as identities. This article began with the understanding that a secure ledger makes creating digital currency straightforward. Let's revisit

this claim. When Alice wishes to pay Bob, she broadcasts the transaction to all bitcoin nodes. A transaction is simply a string: a statement encoding Alice's wish to pay Bob some value, signed by her. The eventual inclusion of this signed statement into the ledger by miners is what makes the transaction real. Note that this doesn't require Bob's participation in any way. But let's focus on what's *not* in the transaction: conspicuously absent are Alice and Bob's identities; instead, the transaction contains only their respective public keys. This is an important concept in bitcoin: public keys are the only kinds of identities in the system. Transactions transfer value from and to public keys, which are called *addresses*.

In order to "speak for" an identity, you must know the corresponding secret key. You can create a new identity at any time by generating a new key pair, with no central authority or registry. You do not need to obtain a user name or inform others that you have picked a particular name. This is the notion of decentralized identity management. Bitcoin does not specify how Alice tells Bob what her pseudonym is—that is external to the system.

Although radically different from most other payment systems today, these ideas are quite old, dating back to David Chaum, the father of digital cash. In fact, Chaum also made seminal contributions to anonymity networks, and it is in this context that he invented this idea. In his 1981 paper, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,"⁹ he states: "A digital 'pseudonym' is a public key used to verify signatures made by the anonymous holder of the corresponding private key."

Now, having message recipients be known only by a public key presents an obvious problem: there is no way to route the message to the right computer. This leads to a massive inefficiency in Chaum's proposal, which can be traded off against the level of anonymity but not eliminated. Bitcoin is similarly exceedingly inefficient compared with centralized payment systems: the ledger containing every transaction is maintained by every node in the system. Bitcoin incurs this inefficiency for security reasons any-



The term *blockchain* has no standard technical definition but is a loose umbrella term used by various parties to refer to systems that bear varying levels of resemblance to bitcoin and its ledger.



way, and thus achieves pseudonymity (that is, public keys as identities) "for free." Chaum took these ideas much further in a 1985 paper,¹¹ where he presents a vision of privacy-preserving e-commerce based on pervasive pseudonyms, as well as "blind signatures," the key technical idea behind his digital cash.

The public-keys-as-identities idea is also seen in b-money and bit gold, the two precursor essays to bitcoin discussed earlier. However, much of the work that built on Chaum's foundation, as well as Chaum's own later work on ecash, moved away from this idea. The cypherpunks were keenly interested in privacy-preserving communication and commerce, and they embraced pseudonyms, which they called *nyms*. But to them, nyms were not mere cryptographic identities (that is, public keys), but rather, usually email addresses that were linked to public keys. Similarly, Ian Goldberg's dissertation, which became the basis of much future work on anonymous communication, recognizes Chaum's idea but suggests that nyms should be human-memorable nicknames with certificates to bind them.²⁰ Thus Bitcoin proved to be the most successful instantiation of Chaum's idea.

The Blockchain

So far, this article has not addressed the blockchain, which, if you believe the hype, is bitcoin's main invention. It might come as a surprise to you that Nakamoto doesn't mention that term at all. In fact, the term *blockchain* has no standard technical definition but is a loose umbrella term used by various parties to refer to systems that bear varying levels of resemblance to bitcoin and its ledger.

Discussing example applications that benefit from a blockchain will help clarify the different uses of the term. First, consider a database backend for transactions among a consortium of banks, where transactions are netted at the end of each day and accounts are settled by the central bank. Such a system has a small number of well-identified parties, so Nakamoto consensus would be overkill. An on-blockchain currency is not needed either, as the accounts are denominated

in traditional currency. Linked time-stamping, on the other hand, would clearly be useful, at least to ensure a consistent global ordering of transactions in the face of network latency. State replication would also be useful: a bank would know that its local copy of the data is identical to what the central bank will use to settle its account. This frees banks from the expensive reconciliation process they must currently perform.

Second, consider an asset-management application such as a registry of documents that tracks ownership of financial securities, or real estate, or any other asset. Using a blockchain would increase interoperability and decrease barriers to entry. We want a secure, global registry of documents, and ideally one that allows public participation. This is essentially what the timestamping services of the 1990s and 2000s sought to provide. Public blockchains offer a particularly effective way to achieve this today (the data itself may be stored off-chain, with only the metadata stored on-chain). Other applications also benefit from a timestamping or “public bulletin board” abstraction, most notably electronic voting.

Let’s build on the asset-management example. Suppose you want to execute trades of assets via the blockchain, and not merely record them there. This is possible if the asset is issued digitally on the blockchain itself, and if the blockchain supports smart contracts. In this instance, smart contracts solve the “fair exchange” problem of ensuring that payment is made if and only if the asset is transferred. More generally, smart contracts can encode complex business logic, provided that all necessary input data (assets, their prices, and so on) are represented on the blockchain.

This mapping of blockchain properties to applications allows us not only to appreciate their potential, but also to inject a much-needed dose of skepticism. First, many proposed applications of blockchains, especially in banking, don’t use Nakamoto consensus. Rather, they use the ledger data structure and Byzantine agreement, which, as shown, date to the 1990s. This belies the claim that blockchains are a new and revolu-



Blockchains are frequently presented as more secure than traditional registries—a misleading claim.



tionary technology. Instead, the buzz around blockchains has helped banks initiate collective action to deploy shared-ledger technology, like the parable of “stone soup.” Bitcoin has also served as a highly visible proof of concept that the decentralized ledger works, and the Bitcoin Core project has provided a convenient code base that can be adapted as necessary.

Second, blockchains are frequently presented as more secure than traditional registries—a misleading claim. To see why, the overall stability of the system or platform must be separated from endpoint security—that is, the security of users and devices. True, the systemic risk of blockchains may be lower than that of many centralized institutions, but the endpoint-security risk of blockchains is far worse than the corresponding risk of traditional institutions. Blockchain transactions are near-instant, irreversible, and, in public blockchains, anonymous by design. With a blockchain-based stock registry, if a user (or broker or agent) loses control of his or her private keys—which takes nothing more than losing a phone or getting malware on a computer—the user loses his or her assets. The extraordinary history of bitcoin hacks, thefts, and scams does not inspire much confidence—according to one estimate, at least 6% of bitcoins in circulation have been stolen at least once.³⁹

Concluding Lessons

The history described here offers rich (and complementary) lessons for practitioners and academics. Practitioners should be skeptical of claims of revolutionary technology. As shown here, most of the ideas in bitcoin that have generated excitement in the enterprise, such as distributed ledgers and Byzantine agreement, actually date back 20 years or more. Recognize that your problem may not require any breakthroughs—there may be long-forgotten solutions in research papers.

Academia seems to have the opposite problem, at least in this instance: a resistance to radical, extrinsic ideas. The bitcoin white paper, despite the pedigree of many of its ideas, was more novel than most academic re-

search. Moreover, Nakamoto did not care for academic peer review and did not fully connect it to its history. As a result, academics essentially ignored bitcoin for several years. Many academic communities informally argued that Bitcoin could not work, based on theoretical models or experiences with past systems, despite the fact it *was* working in practice.

We have seen repeatedly that ideas in the research literature can be gradually forgotten or lie unappreciated, especially if they are ahead of their time, even in popular areas of research. Both practitioners and academics would do well to revisit old ideas to glean insights for present systems. Bitcoin was unusual and successful not because it was on the cutting edge of research on any of its components, but because it combined old ideas from many previously unrelated fields. This is not easy to do, as it requires bridging disparate terminology, assumptions, and so on, but it is a valuable blueprint for innovation.

Practitioners would benefit from being able to identify overhyped technology. Some indicators of hype: difficulty identifying the technical innovation; difficulty pinning down the meaning of supposedly technical terms, because of companies eager to attach their own products to the bandwagon; difficulty identifying the problem that is being solved; and finally, claims of technology solving social problems or creating economic/political upheaval.

In contrast, academia has difficulty selling its inventions. For example, it's unfortunate that the original proof-of-work researchers get no credit for bitcoin, possibly because the work was not well known outside academic circles. Activities such as releasing code and working with practitioners are not adequately rewarded in academia. In fact, the original branch of the academic proof-of-work literature continues today without acknowledging the existence of bitcoin! Engaging with the real world not only helps get credit, but will also reduce reinvention and is a source of fresh ideas.

Acknowledgments. Thanks to Adam Back, Andrew Miller, Edward Felten, Harry Kalodner, Ian Goldberg, Ian

Grigg, Joseph Bonneau, Malte Möser, Mike Just, Neha Narula, Steven Goldfeder, and Stuart Haber for their valuable feedback. □

References

- Aspnes, J., et al. Exposing computationally challenged Byzantine imposters. Yale University Department of Computer Science, 2005; <http://cs.yale.edu/publications/techreports/tr1332.pdf>.
- Back, A. A partial hash collision based postage scheme, 1997; <http://www.hashcash.org/papers/announce.txt>.
- Back, A. Hash cash, 2001; <https://web.archive.org/web/20010614013848/http://cypherspace.org/hashcash/>.
- Back, A. Hashcash—a denial of service counter measure, 2002; <http://www.hashcash.org/papers/hashcash.pdf>.
- Bayer, D., Haber, S. and Stornetta, W.S. Improving the efficiency and reliability of digital time-stamping. In *Proceedings of Sequences* (1991); https://link.springer.com/chapter/10.1007/978-1-4613-9323-8_24.
- Benaloh, J., de Mare, M. Efficient broadcast timestamping, 1991; <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.9199>.
- Boyle, T.F. GLT and GLR: Component architecture for general ledgers, 1997; <https://linas.org/mirrors/www.gldialtone.com/2001.07.14/GLT-GLR.htm>.
- Castro, M. and Liskov, B. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (1999); <http://pmg.csail.mit.edu/papers/osdi99.pdf>.
- Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (Feb. 1981), 84–90; <https://dl.acm.org/citation.cfm?id=358563>.
- Chaum, D. Blind signatures for untraceable payments. *Advances in Cryptology*, 1983, 199–203.
- Chaum, D. Security without identification: transaction systems to make Big Brother obsolete. *Commun. ACM* 28, 10 (Oct. 1985), 1030–1044; <https://dl.acm.org/citation.cfm?id=4373>.
- Chaum, D., et al. Untraceable electronic cash. *Advances in Cryptology*, 1988, 319–327; <https://dl.acm.org/citation.cfm?id=88969>.
- Dai, W. 1998; <http://www.weidai.com/bmoney.txt>.
- Douceur, J.R. The Sybil attack, 2002; <https://dl.acm.org/citation.cfm?id=687813>.
- Dwork, C. and Naor, M. Pricing via processing or combatting junk mail, 1992; <https://dl.acm.org/citation.cfm?id=705669>.
- Felten, E. Smart contracts: neither smart nor contracts? Freedom to tinker, 2017; <https://freedom-to-tinker.com/2017/02/20/smart-contracts-neither-smart-not-contracts/>.
- Franklin, M.K. and Malkhi, D. Auditible metering and lightweight security, 1997; <http://www.hashcash.org/papers/auditible-metering.pdf>.
- Gabber, E., et al. Curbing junk e-mail via secure classification, 1998; <http://www.hashcash.org/papers/secure-classification.pdf>.
- Garay, J.A., et al. The bitcoin backbone protocol: analysis and applications. *Advances in Cryptology*, 2015, 281–310; <https://eprint.iacr.org/2014/765.pdf>.
- Goldberg, I. A pseudonymous communications infrastructure for the Internet. Ph.D. dissertation. University of California Berkeley, 2000; <http://morla.freehaven.net/anonbib/cache/ian-thesis.pdf>.
- Grigg, I. Triple entry accounting, 2005; http://iang.org/papers/triple_entry.html.
- Haber, S. and Stornetta, W.S. How to timestamp a digital document. *J. Cryptology* 3, 2 (1991), 99–111; https://link.springer.com/chapter/10.1007/3-540-38424-3_32.
- Haber, S. and Stornetta, W.S. Secure names for bit-strings. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, 28–35; <http://dl.acm.org/citation.cfm?id=266430>.
- Jakobsson, M. and Juels, A. Proofs of work and bread pudding protocols, 1999; <http://www.hashcash.org/papers/bread-pudding.pdf>.
- Juels, A. and Brainard, J. Client puzzles: a cryptographic countermeasure against connection completion attacks. In *Proceedings of Networks and Distributed Security Systems*, 1999, 151–165; <https://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/juels.pdf>.
- Just, M. Some timestamping protocol failures, 1998;

<http://www.isoc.org/isoc/conferences/ndss/98/just.pdf>.

- Lamport, L., et al. The Byzantine Generals Problem. *ACM Trans. Programming Languages and Systems* 4, 3 (1982), 382–401; <https://dl.acm.org/citation.cfm?id=357176>.
- Lamport, L. The part-time parliament. Digital Equipment Corp., 1989; https://computerarchive.org/files/mirror/www.bitsavers.org/pdf/dec/tech_reports/SRC-RR-49.pdf.
- Lamport, L. Paxos made simple, 2001; <http://lamport.azurewebsites.net/pubs/paxos-simple.pdf>.
- Laurie, B. Certificate transparency. *acmqueue* 12, 1 (2014); <https://queue.acm.org/detail.cfm?id=2668154>.
- Levy, K.E.C. Book-smart, not street-smart: blockchain-based smart contracts and the social workings of law. *Engaging Science, Technology, and Society* 3 (2017), 1–15; <http://estsjournal.org/article/view/107>.
- Melara, M., et al. CONIKS: Bringing key transparency to end users. In *Proceedings of the 24th Usenix Security Symposium*, 2015; <https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-melara.pdf>.
- Merkle, R.C. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1980; <http://www.merkle.com/papers/Protocols.pdf>.
- Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system, 2008; <https://bitcoin.org/bitcoin.pdf>.
- Nakamoto, S. Re: Bitcoin P2P e-cash paper, 2008; <http://satoshi.nakamotoinstitute.org/emails/cryptography/11/>.
- Narayanan, A., et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016; <http://bitcoinbook.cs.princeton.edu/>.
- Pass, R., et al. Analysis of the blockchain protocol in asynchronous networks. In *Proceedings of the 2017 International Conference on the Theory and Applications of Cryptographic Techniques*; https://link.springer.com/chapter/10.1007/978-3-319-56614-6_22.
- Pinkas, B. and Sander, T. Securing passwords against dictionary attacks. In *Proceedings of the Ninth ACM Conference on Computer and Communications Security*, 2002, 161–170; <https://dl.acm.org/citation.cfm?id=586133>.
- Reuters. Mind your wallet: Why the underworld loves bitcoin, 2014; <http://www.cnbc.com/2014/03/14/mind-your-wallet-why-the-underworld-loves-bitcoin.html>.
- Rivest, R.L. and Shamir, A. PayWord and MicroMint: Two simple micropayment schemes. In *Proceedings of the 1996 International Workshop on Security Protocols*.
- Sirer, E.G. Bitcoin guarantees strong, not eventual, consistency. *Hacking, Distributed*, 2016; <http://hackingdistributed.com/2016/03/01/bitcoin-guarantees-strong-not-eventual-consistency/>.
- Szabo, N. Smart contracts, 1994; <http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>.
- Szabo, N. Bit gold. Unenumerated, 2008; <https://unenumerated.blogspot.com/2005/12/bit-gold.html>.
- Vishnumurthy, S., Chandrakumar, S. and Sirer, E.G. Karma: A secure economic framework for P2P resource sharing. In *Proceedings of the Workshop on the Economics of Peer-to-Peer Systems* (Berkeley, CA, June 2003).
- Wattenhofer, R. *The Science of the Blockchain*. Inverted Forest Publishing, 2016.

Arvind Narayanan is an assistant professor of computer science at Princeton University. He leads the Princeton Web Transparency and Accountability Project to uncover how companies collect and use our personal information. Twitter @random_walker.

Jeremy Clark is an assistant professor at the Concordia Institute for Information Systems Engineering. He has also worked with several municipalities on voting technology and testified to the Canadian Senate on bitcoin. Twitter @PulpSpY.

Copyright held by owners/authors.

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

Cardboard surrounds and protects stuff as it crosses boundaries.

BY PAT HELLAND

XML and JSON Are Like Cardboard

IN TODAY'S WORLD, cardboard is an ever-important part of life. Given the major investment of resources and money, you might question whether it's worth it. It turns out the efficiencies and savings from cardboard outstrip the costs to manufacture and later recycle it.

Semi-structured representations of data are not the cheapest format. There is typically a lot of extra stuff like angle brackets contained in it. JSON, XML, and other semi-structured representations allow for wonderful flexibility and dynamic interpretation. The efficiencies and savings gained from flexibility more than make up for the overhead.

Cardboard surrounds and protects stuff as it moves across boundaries. No one uses cardboard to move parts around within a factory. Instead, they use custom-designed containers that are specially purposed for the parts being produced. Cardboard is used to protect the stuff as it leaves the factory.

JSON and XML are used to protect data when it moves across trust boundaries. Semi-structured data wraps a single message or a single item in a key-value store in a way that allows for flexibility and extensibility. Inside an application, relational data is more tightly controlled and well formed. Evolving your relational data inside the trust and management boundary of an app is tractable.

SQL and relational data are easier and better for processing data *within a trust boundary*. XML and JSON are more flexible and dynamic as they capture the information and its metadata. This makes it easy and flexible to squirt data across trust boundaries.

Self-defining and self-identifying. Cardboard is usually self-describing. Your new TV has printing on the outside of the box telling you what's inside the box. As you move your old TV to your new home, you write "TV" on the outside of the moving box.

JSON and XML are usually self-describing. This can be done by referencing a schema or by examining the attributes expressed within the document/file itself.

Generic vs. custom. The last time we moved to a new home, I bought a bunch of boxes of varying sizes, tape, wrapping paper, and padding, and a bunch of marking pens. Like most other folks in the throes of packing and moving, we worked hard to describe the contents of every box we filled, but we occasionally messed up and omitted some items from the list as everything went into a box. Most things fit well into one of the standard boxes, although some of our household items involved really creative uses of cardboard, tape, and padding as we worked to protect our stuff.

Manufactured items frequently have custom-made cardboard protection. My wife loves the vacuum cleaners from one particular manufacturer. Indeed, the shape, form, and workings of the vacuums can be fun and surprising. To me, half the fun is disassembling the cardboard protection used inside the cardboard box. There are dozens of spe-



cial pieces of cardboard wedged into every nook and cranny of the vacuum. Man, that vacuum is well protected! I suspect they have a factory just to create the specialized pieces of cardboard. I also suspect the savings from avoiding damage are well worth it.

XML grew out of the document markup world. It descended from SGML (Standard Generalized Markup Language), which was originally intended to separate the text of a document from its formatting. XML is very strongly oriented around letting you “do your own thing” with the format.

Yet, on top of the flexible “do your own thing” approach, there are mechanisms to impose rigor and constraints on XML documents. XML Schema came into being in the early 2000s as a means of ensuring consistency for a set of messages. A document is validated if it conforms to an XML schema definition. In this way, some usages of XML are constrained to fit a particular shape and form.

One of the wonderful things about XML and JSON is their flexibility. In some applications, they support a tightly prescribed schema much like the cardboard protecting the vacuum cleaner. In other applications, they allow you to toss in all your family goods, including the kitchen sink. Sometimes, there is a tightly prescribed schema of required data while the sender can toss in extensions to its heart’s content.

Crossing boundaries. In general, semi-structured data is used to cross boundaries in your computing environment. Documents containing human-readable stuff are kept on websites. REST calls are made across services that may or may not reside within the same company.

The loose coupling of semi-structured data allows the sending and receiving services to evolve separately with much lower friction. Changing tightly coupled stuff requires coordination that is just plain difficult.

Crossing boundaries with key-value stores. Frequently, semi-structured data in documents or files is stored in a file system or a key-value store. It is valuable to have readers and writers of these docs/files decoupled in their metadata. To have the shape and form of the data described in the contents of the docs and files makes it possible to evolve the various users with less friction than you would see if the metadata were strict and rigid. This is why we see the success of semi-structured representations for stored stuff.

It’s not the size that counts. It turns out the weight and size of the cardboard are not that big of a deal. You have surely had the experience of receiving some small item such as a computer chip packaged in a box that weighs a *lot* more than the stuff being protected. It makes economic sense to protect the tiny thing well.

Large e-commerce sites ship tens of thousands of different things of different sizes. Still, they find it more efficient to use a relatively small number of box sizes. Consequently, it’s common to open the box and find a tiny thing and a whole bunch of padding.

Similarly, you shouldn’t be too worried about the bulkiness of your files and documents. The embedded metadata can take a lot of space. Lord knows, an XML file has a lot of angle brackets! Still, the value accrued from the features of semi-structured data is worth it. As long as the world doesn’t run out of angle brackets, it will be all right.

Gotta take care of your stuff! In cardboard, the safety and care for stuff is the important reason for its existence. Similarly, in XML and JSON the safety and care of the data, both in transit and in storage, are why we bother.

Now, if only we could figure out efficient recycling for used angle brackets, we would be good to go ...

Related articles on queue.acm.org

The Power of Babble

Pat Helland

<http://queue.acm.org/detail.cfm?id=3003188>

Rules for Mobile Performance Optimization

Tammy Everts

<http://queue.acm.org/detail.cfm?id=2510122>

Schema.org: Evolution of Structured Data on the Web

R.V. Guha, Dan Brickley, and Steve Macbeth

<http://queue.acm.org/detail.cfm?id=2857276>

Pat Helland has been implementing transaction systems, databases, application platforms, distributed systems, fault-tolerant systems, and messaging systems since 1978. He currently works at Salesforce.

Copyright held by author/owner.
Publications rights licensed to ACM. \$15.00.



Article development led by **acmqueue**
queue.acm.org

**Expert-curated guides to
the best of CS research.**

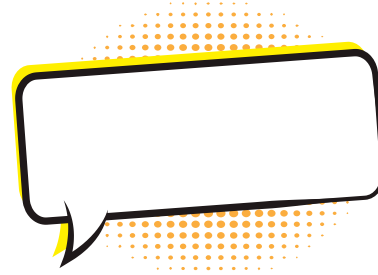
BY JOHN REGEHR

Research for Practice: Vigorous Public Debates in Academic Computer Science

THIS INSTALLMENT OF Research for Practice features a special curated selection from John Regehr, who takes us on a tour of great debates in academic computer science research. In case you thought flame wars were reserved for Usenet mailing lists and Twitter, think again: the academic literature is full of dramatic, spectacular, and vigorous debates spanning file systems, operating system kernel design, and formal verification. Please enjoy!

—Peter Bailis

Peter Bailis is an assistant professor of computer science at Stanford University. His research in the Future Data Systems group (futuredata.stanford.edu) focuses on the design and implementation of next-generation data-intensive systems.



Science is often portrayed as a world of cold logic and hard facts, but as human beings we have great difficulty keeping emotion and prejudice

out of the picture—whether the issue at hand is inconsequential (“Is Pluto a planet?”) or threatening to our existence (“Can we stop global warming?”). Historically, as Galileo’s encounters with the Roman Inquisition showed, unorthodoxy could have very serious consequences. Recent scientific debates have tended to be calmer, but being on the wrong side of an issue can still have career-altering consequences (<http://bit.ly/2xzYddN>). Computer science, perhaps because it is young and well funded, seems to have been relatively free of real schisms, but it still inspires energetic debates.

Computer science does not have a culture of retraction, and in any case many of these debates are not about the kinds of mistakes that lead to retractions. The useful debates, the ones we can learn from, are those where both sides are publicly available in writing. These are a valuable instructional resource, and I sometimes assign them as reading to my students. They show an important part of science that often gets swept under the rug—students enjoy seeing that things are not as cut-and-dried as teachers often try to make them sound. It is useful to try to figure out who is right and who is wrong. Alternatively, some debates are more about personalities, and still others feature researchers talking past each other based on their different assumptions and perspectives.

Considered harmful. An early debate, which feels a bit quaint now, began when Edsger Dijkstra published Go-To Statement Considered Harmful (1968), which argued against unstructured control flow in programming languages. Follow-ups included Go-To Considered Harmful Considered Harmful and Go-To Considered Harmful Considered Harmful Considered Harmful (1987).

- ▶ <http://bit.ly/2wZp831>
- ▶ <http://bit.ly/1SbpgAr>
- ▶ <https://dl.acm.org/citation.cfm?doid=22899.315729>

Multiple versions of the facts. One of my favorite public debates concerns *N-version* programming: a software-development method where several implementations of a specification are run in parallel and voting is used to determine the correct result. If independent implementations have independent defects, this method would lead to a substantial increase in software reliability. John C. Knight and Nancy G. Leveson wrote a paper (1986) showing that the assumption of independent faults is suspect. This finding did not sit well with the proponents of *N-version* programming, and while I cannot find online copies of their rebuttals, Knight and Leveson's reply to the criticisms includes plenty of quotes. This is great reading, a classic of the genre.

- ▶ <http://sunnyday.mit.edu/papers/nver-tse.pdf>
- ▶ <http://sunnyday.mit.edu/critics.pdf>

Can we at least agree that CATOCS is a great acronym? Starting in the late 1980s, Ken Birman's group was advocating causal and totally ordered multicast: A primitive for building distributed systems that provides strong guarantees about internode communication in distributed systems. David Cheriton and Dale Skeen were less than impressed and wrote 15 pages to that effect (1993). Birman wrote a long response to the criticisms. Also see Neha Narula's later take on the debate (2013).

- ▶ <http://www.cs.cornell.edu/courses/cs614/2003sp/papers/BSS91.pdf>
- ▶ https://www.cs.rice.edu/~alc/comp520/papers/Cheriton_Skeen.pdf
- ▶ <http://bit.ly/2hCDXCc>
- ▶ <http://dsrg.pdos.csail.mit.edu/2013/06/13/cheriton-and-skeen/>

File system performance evaluation is difficult. A 1991 paper introduced log-based file systems, which increase the performance of writes to files by reducing the number of seeks needed to perform a file update. In 1993 Margo

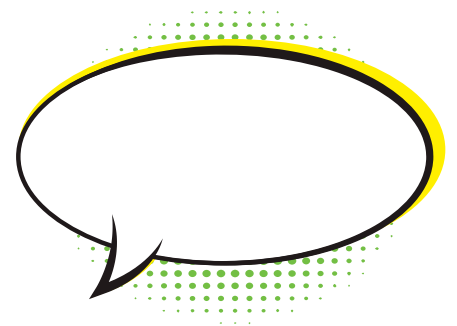
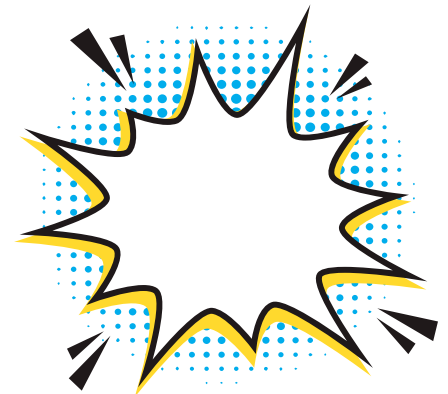
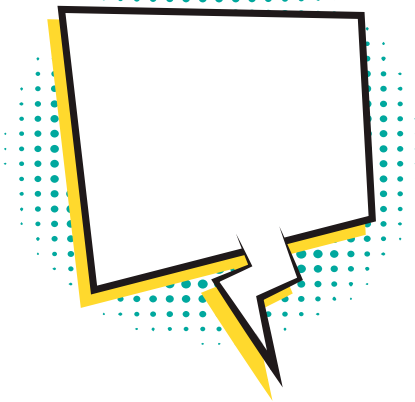
Seltzer et al. published a paper describing and evaluating an implementation of a log-based file system, with a follow-up in 1995. John Ousterhout, one of the authors of the original paper, disagreed with the evaluation. Seltzer and her coauthors rebutted his critique, and Ousterhout had, as far as I know, the last word.

- ▶ <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.1363∓rep=rep1&type=pdf>
- ▶ <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/usenix-winter95.pdf>

▶ <https://www.usenix.org/legacy/publications/library/proceedings/sd93/seltzer.pdf>

- ▶ http://www.eecs.harvard.edu/~margo/papers/usenix95-lfs/supplement/ouster_critique1.html
- ▶ <http://www.eecs.harvard.edu/~margo/papers/usenix95-lfs/supplement/rebuttal.html>
- ▶ http://www.eecs.harvard.edu/~margo/papers/usenix95-lfs/supplement/ouster_critique2.html

You would not get a high grade for such a design. Another classic debate, Torvalds vs. Tanenbaum (1992), was



about how operating systems should be structured: as a monolithic collection of code running in kernel mode, or instead as a group of independent subsystems isolated by the memory management unit. Also see some (one-sided) comments on a reincarnation of the debate. Related to this discussion, in 2005, Steven Hand et al. published “Are Virtual Machine Monitors Microkernels Done Right?” In response, Gernot Heiser et al. wrote a paper with the same title in 2006 but coming to the opposite conclusion.

► <http://www.oreilly.com/openbook/opensources/book/appa.html>

► https://www.usenix.org/legacy/event/hotos05/final_papers/full_papers/hand/hand.pdf

► <http://cgi.di.uoa.gr/%7Emema/courses/mde518/papers/heiser.pdf>

A very obnoxious paper? “Social Processes and Proofs of Theorems and Programs” is a provocative opinion piece written in 1979 by Richard De Millo et al. about the role of formal methods in software development. Dijkstra called it “a very obnoxious paper” (see p. 14 of a transcript of an interview with Dijkstra from 2001) and wrote a response called “A Political Pamphlet from the Middle Ages.” De Millo et al. replied: “We must begin by refusing to concede that our confidence in a piece of real software has ever been increased by a proof of its correctness...” See also *Communications’* Letters to the Editor responding to this article, Victor Yodaiken’s take on the debate, and three more shots fired in 2010—two by Moshe Vardi and one by the original paper’s authors.

► http://www.yodaiken.com/papers/p271-de_millo.pdf

► <http://conservancy.umn.edu/bitstream/handle/11299/107247/oh330ewd.pdf>


► <http://www.cs.utexas.edu/users/EWD/transcriptions/EWD06xx/EWD638.html>

► <https://conservancy.umn.edu/bitstream/handle/11299/107247/oh330ewd.pdf>

► <http://research.microsoft.com/en-us/um/people/lamport/pubs/letter-to-editor.pdf>

► <http://www.yodaiken.com/2008/11/10/dijkstra-versus-perlis/>

► <http://cacm.acm.org/>



“Social Processes and Proofs of Theorems and Programs” is a provocative opinion piece written in 1979 by De Millo et al. about the role of formal methods in software development. Dijkstra called it “a very obnoxious paper.”



[magazines/2010/1/55739-more-debate-please/fulltext](#)

This guy’s arrogance takes your breath away. Dijkstra and John Backus had an (only partially public) spat in the late 1970s.

► <https://medium.com/%40acidflask/this-guys-arrogance-takes-your-breath-away-5b903624ca5f>

SWATT or be SWATTed. The computer security research community has an especially strong tradition of refuting published results. For example, SWATT (software-based attestation) offers a protocol for checking that a remote system has the memory image it is supposed to have. A 2009 paper called “On the Difficulty of Software-based Attestation of Embedded Devices” presents concrete attacks on SWATT. SWATT authors Adrian Perrig and Leendert van Doorn did not agree that the attacks were valid, and, finally, the paper’s authors, Aurelian Francillon et al., responded to the refutation.

► <http://www.netsec.ethz.ch/publications/papers/swatt.pdf>

► <https://pdfs.semanticscholar.org/fe14/909505a02a484811ff70ccb326905f352d0a.pdf>

► <http://www.netsec.ethz.ch/publications/papers/perrig-ccs-refutation.pdf>


► <https://pdfs.semanticscholar.org/657a/7b4270581c655763df0f5a1ddb79cb7cb946.pdf>

A matter of integrity. Code-pointer integrity (CPI) is a technique for avoiding control-flow hijacking caused by memory safety errors in C or C++ code. Missing the Point(er) (2015) presents attacks against CPI, while Getting the Point(er) (2015) argues in favor of the security of CPI.

► <http://dslab.epfl.ch/pubs/cpi.pdf>

► <https://people.csail.mit.edu/rinard/paper/oakland15.pdf>

► <http://dslab.epfl.ch/pubs/cpi-getting-the-pointer.pdf>

Acknowledgments. I’d like to thank many blog readers and Twitter users for providing feedback on the original blog post from which this article was derived. 

John Regehr is a computer science professor at the University of Utah. He likes to create software tools for making software better.

Copyright held by owner/author.
Publication rights licensed to ACM. \$15.00.

**Previous
A.M. Turing Award
Recipients**

1966 A. J. Perlis
1967 Maurice Wilkes
1968 R.W. Hamming
1969 Marvin Minsky
1970 J.H. Wilkinson
1971 John McCarthy
1972 E.W. Dijkstra
1973 Charles Bachman
1974 Donald Knuth
1975 Allen Newell
1975 Herbert Simon
1976 Michael Rabin
1976 Dana Scott
1977 John Backus
1978 Robert Floyd
1979 Kenneth Iverson
1980 C.A.R Hoare
1981 Edgar Codd
1982 Stephen Cook
1983 Ken Thompson
1983 Dennis Ritchie
1984 Niklaus Wirth
1985 Richard Karp
1986 John Hopcroft
1986 Robert Tarjan
1987 John Cocke
1988 Ivan Sutherland
1989 William Kahan
1990 Fernando Corbató
1991 Robin Milner
1992 Butler Lampson
1993 Juris Hartmanis
1993 Richard Stearns
1994 Edward Feigenbaum
1994 Raj Reddy
1995 Manuel Blum
1996 Amir Pnueli
1997 Douglas Engelbart
1998 James Gray
1999 Frederick Brooks
2000 Andrew Yao
2001 Ole-Johan Dahl
2001 Kristen Nygaard
2002 Leonard Adleman
2002 Ronald Rivest
2002 Adi Shamir
2003 Alan Kay
2004 Vinton Cerf
2004 Robert Kahn
2005 Peter Naur
2006 Frances E. Allen
2007 Edmund Clarke
2007 E. Allen Emerson
2007 Joseph Sifakis
2008 Barbara Liskov
2009 Charles P. Thacker
2010 Leslie G. Valiant
2011 Judea Pearl
2012 Shafi Goldwasser
2012 Silvio Micali
2013 Leslie Lamport
2014 Michael Stonebraker
2015 Whitfield Diffie
2015 Martin Hellman
2016 Sir Tim Berners-Lee

ACM A.M. TURING AWARD NOMINATIONS SOLICITED

Nominations are invited for the 2017 ACM A.M. Turing Award. This is ACM's oldest and most prestigious award and is given to recognize contributions of a technical nature which are of lasting and major technical importance to the computing field. The award is accompanied by a prize of \$1,000,000. Financial support for the award is provided by Google Inc.

Nomination information and the online submission form are available on:
http://amturing.acm.org/call_for_nominations.cfm

Additional information on the Turing Laureates is available on:
<http://amturing.acm.org/byyear.cfm>

The deadline for nominations/endorsements is January 15, 2018.

For additional information on ACM's award program please visit: www.acm.org/awards/



Association for
Computing Machinery

Cyber deterrence, like nuclear deterrence, depends on our adversaries being rational enough to be deterred by our threats but not by theirs.

BY MARTIN E. HELLMAN

Cybersecurity, Nuclear Security, Alan Turing, and Illogical Logic

THE 2015 ACM A.M. Turing Award recognized work I did 40 years ago, so it is understandable that my interests have changed significantly, with my most recent project being a book, *A New Map for Relationships: Creating True Love at Home & Peace on the Planet*, co-authored with my wife Dorothea. While, at first glance, the book might seem to have nothing in common with my work on cryptography, my Turing Lecture drew a number of parallels I will bring out in what follows.

The story starts in March 1975, when the U.S. National Bureau of Standards (NBS), now known as

the National Institute of Standards and Technology (NIST), proposed a Data Encryption Standard (DES) to protect unclassified but sensitive data. Whitfield Diffie, with whom I shared the Award, and I quickly realized that DES's 56-bit key size was inadequate and needed to be increased.

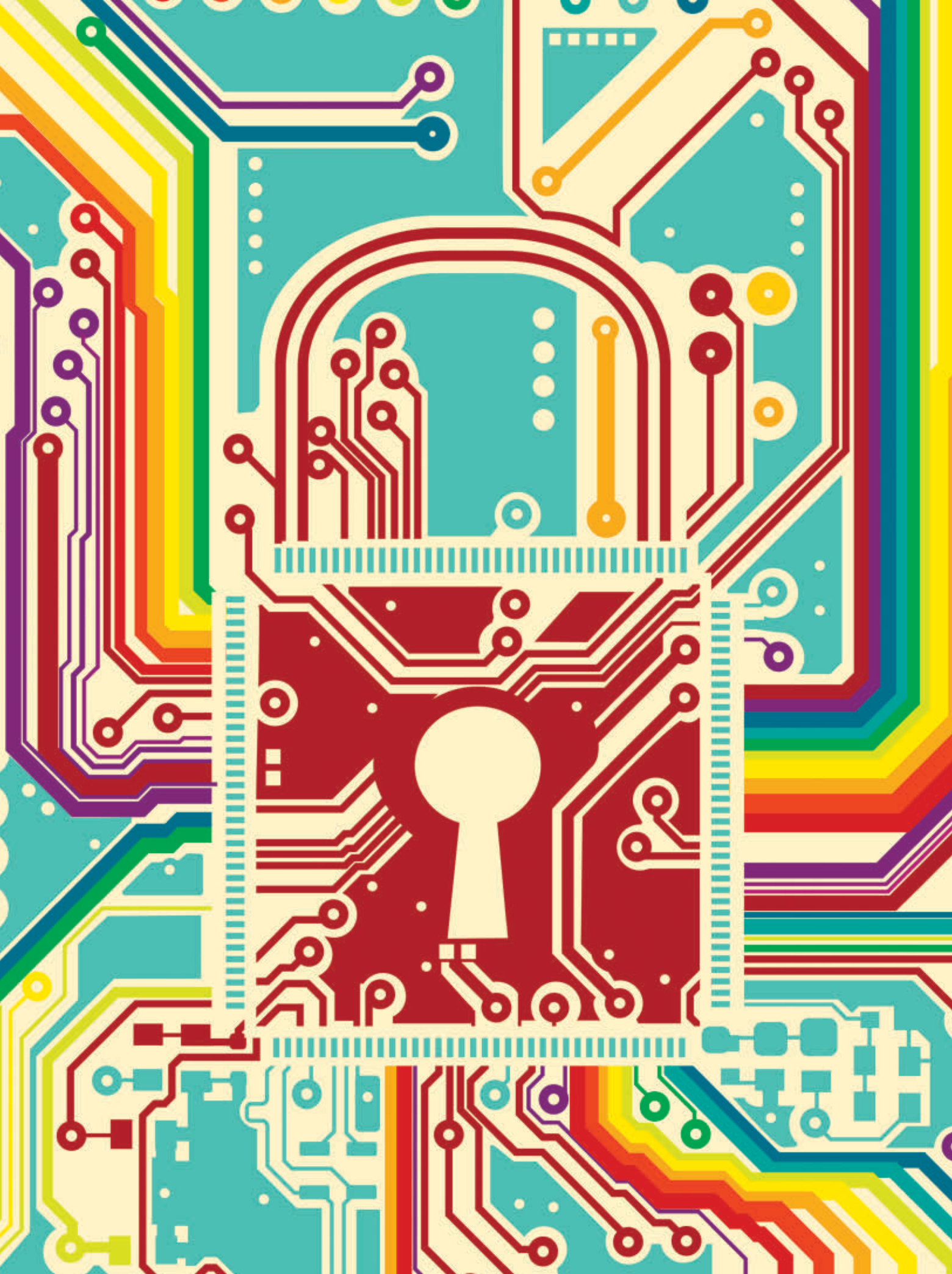
DES had 2^{56} , or approximately 10^{17} , keys. We estimated that the 1975 technology would allow a single-chip search engine to check 10^6 keys per second, so 10^6 such chips could search the entire key space in 10^5 seconds. That is approximately one day, and we estimated the equivalent cost to be on the order of \$5,000 per recovered key. We also noted that the decreasing cost of computation—roughly a factor of 10 every five years—would rapidly reduce this cost. Even an order-of-magnitude error in our estimate would thus be erased in a short time.³

We initially thought the inadequate key size was a mistake that would be corrected once we pointed it out, but NBS resisted, claiming our estimates were off by four orders of magnitude. Our initial estimate had been a rough order-of-magnitude approximation that was adequate to show the need for an increased key size. But NBS's estimate was clearly wrong, and we came to realize we were indirectly battling the National Security Agency (NSA), in addition to NBS.

A larger key size would allow foreign governments, criminals, and terrorists to hide their communications from NSA, while 56 bits would not. What we had thought was a technical problem

» key insights

- While revolutionary, public key cryptography can also be viewed as a natural step in the evolution of the field of cryptography.
- There is greater risk than is generally recognized that a major advance in factoring and discrete logarithms might break existing public key systems.
- In making ethical decisions, we need to zealously guard against fooling ourselves about our real motivations.



turned out to be political. If we wanted to improve the security of the standard, we would have to treat it as a political battle by seeking media coverage and Congressional hearings—which we did.

The fight that followed was part of “the first crypto war.” While the media and several members of Congress supported Diffie’s and my position, we lost this part of it. DES, including its 56-bit key, was the official encryption standard from 1977 until 2002 when it was superseded by the Advanced Encryption Standard, or AES, which has a minimum key size of 128 bits.

Diffie and I recommended triple-DES³ as a simple, albeit more expensive, way to improve DES security, but most implementations used the less-secure approach.

Public Key Cryptography and the DES Controversy

Within a year of DES being proposed in 1975, a development—the invention of public key cryptography by Diffie and me⁴ and independently by Ralph Merkle¹²—exacerbated NSA’s concerns.

While Diffie and I saw a 56-bit key as small, we now know it looked large from NSA’s perspective. Prior to DES, most commercial encryption systems could be broken much faster than DES, and most data was sent unencrypted, allowing access at no cryptanalytic cost.

In comparison, even \$5,000 per recovered key was a huge impediment to NSA’s communications-intelligence operation. But it appears to have reasoned that cost would limit the frequency of key changes so a recovered key would be useful for months, perhaps years. The invention of public key cryptography allowed keys to be changed as frequently as desired, making \$5,000 per key a much more daunting barrier for an adversary.

Evolution of Public Key Cryptography

While public key cryptography is seen as revolutionary—a characterization I love—after the following explanation, one might wonder why it took Diffie, Merkle, and me so long to discover.

Diffie and I had been talking about “trapdoor cryptosystems” (TDCs) for some time before we devised the public key concept, and it is but a small step



While Diffie and I saw a 56-bit key as small, we now know it looked large from NSA’s perspective.



from TDCs to public key.⁴ TDCs occurred to us because, in the military, you want a highly secure cipher for use by your own troops but do not want it to be used to keep secrets from you if it is captured by your adversary. We realized that a solution was to build trapdoor information into the cryptosystem that would allow the designer to break it easily if it was used against him, but without that information his adversary would be unable to cryptanalyze his encrypted messages. While we never developed a workable TDC, the concept figured prominently in a later analysis of DES Diffie and I undertook, with others.⁸ We found structures within DES that looked like they might constitute a trapdoor, although later developments indicate they were probably due to efforts to strengthen the algorithm against differential cryptanalysis.¹

It is also noteworthy that half of the public key concept—public key exchange—occurred independently to three different groups within a short period of time.

According to documents declassified years later,⁵ variations occurred in 1970, 1973, and 1974 to researchers James Ellis, Clifford Cocks, and Malcolm Williamson of the Government Communications Headquarters (GCHQ), the British agency responsible for providing signals intelligence and information assurance to that nation, though none of their work envisioned digital signatures.

Ralph Merkle, then a student at the University of California at Berkeley, developed the concept of a public key distribution system in the fall of 1974 and published it, along with a proof of concept (“Merkle puzzles”), in *Communications*, April 1978.¹²

Unaware of the still-secret GCHQ work and Merkle’s budding ideas, Diffie and I proposed a more general framework—a public key cryptosystem—in the Spring of 1975. This approach included digital signatures, as well as public key exchange, with digital signatures being an entirely new idea, even within the classified community.

In May 1976, Diffie and I developed the first practical, unclassified system for public key exchange, publishing both it and the public key cryptosystem concept in our paper “New Directions in Cryptography” in *IEEE Transactions*

on *Information Theory*, November 1976.⁴ That public key exchange system is widely known as Diffie-Hellman Key Exchange, but somewhat ironically, it is an implementation of Merkle's public key distribution system concept, not our public key cryptosystem concept. I therefore refer to it as the "Diffie-Hellman-Merkle Key Exchange."

In light of the frequent interactions Diffie and I had, I regard everything in "New Directions" as joint work, though some scholars have noted (correctly) that Diffie devised the public key cryptosystem concept, while I discovered the Diffie-Hellman-Merkle Key Exchange algorithm. Because those individual insights were based on long-term joint work, I tend not to separate credit.

A full, working public key cryptosystem was not realized until April 1977 when Ron Rivest, Adi Shamir, and Leonard Adleman published their MIT report that, in slightly modified form, became their famous 1978 "RSA paper" in *Communications*.¹⁷

While Merkle's 1978 publication date—two years after "New Directions"—gives the impression that it followed in our footsteps, he submitted his paper earlier than we did, in August 1975. Its publication was delayed by an editor who initially rejected it, writing, on October 22, 1975, "I ... was particularly bothered by the fact that there are no references to the literature. Has anyone else ever investigated this approach?"⁶

In the editor's defense, Merkle was a student unfamiliar with how to write and adequately reference a technical paper; the person who reviewed it (described by the editor as "an experienced cryptography expert") recommended against publishing it, noting that it "... is not in the mainstream of present cryptography thinking," and no one else at Berkeley, where Merkle was then a student, appreciated his work. Earlier, in the fall of 1974, a Berkeley professor discouraged him from pursuing public key distribution as a term project, telling him, "Project 2 [a much more mundane proposal] looks more reasonable, maybe because your description of Project 1 is muddled terribly." Merkle dropped the course and pursued public key distribution on his own.

Born Classified?

NSA's concerns led it to try to control dissemination of our work.² In January 1976, soon after Diffie and I realized the need to treat DES's inadequate key size as a political rather than a technical problem, two high-level NSA employees flew out to California and tried to dissuade us from pursuing the matter. They basically told us, "You're wrong, but please be quiet. If you keep talking this way, you will cause grave harm to national security." But that did not compute. What they were really saying was, "You're right, but please be quiet. If you keep talking this way, you will cause grave harm to national security."

I went home that evening to decide the right thing to do. NSA was telling me the right thing was to be quiet, while my intellect told me the opposite, even from a purely national perspective. The U.S. was the world's most computerized nation, with the most to lose from insecure encryption. The Soviet Union had much less to lose and much more to gain from leaving the DES key at 56 bits. Also, NSA's request occurred soon after the Watergate revelations had shown that claims of national security could be misused to the detriment of the nation.

As I was trying to decide the right thing to do, an idea popped into my head: "Forget about what is right and wrong. You have a tiger by the tail and will never have as much chance to influence events. Run with it!"

Somehow, what would normally be an unconscious "shadow motivation" had managed to bubble to the surface and become a "devil on my shoulder," like in the movies. At the time, I thought I had brushed the devil off my shoulder and made a rational decision to go public with our analysis of the standard's weakness. But five years later, in trying to understand the motivation of the Manhattan Project scientists who developed the atom bomb during World War II, I realized I had fooled myself. Instead of doing what was right, I had figured out what I wanted to do, and then had come up with the rationalization for doing it.

I was fortunate that my decision to go public was the right one, even though I had fooled myself about my motivation. But that was sheer luck. If I had been working on the Manhattan

Project, the consequences of fooling myself would have been far more grave, I vowed never to fool myself again, although implementing that decision proved tricky during Stanford University's patent fight with RSA Data Security. Space does not allow me to provide the details here, but the interested reader can find a description on pages 46–54 of our book;⁷ a free .pdf file is also available at <http://tinyurl.com/HellmanBook>, expanding to <http://www-ee.stanford.edu/%7Ehellman/publications/book3.pdf>. Those same pages explain why I believe the Manhattan Project scientists fooled themselves about their motivation for working on the bomb.

The fight Diffie and I were having with NSA came to a head on July 7, 1977, when one of its employees wrote to the IEEE, claiming it was breaking the law by publishing our papers.¹⁴ He cited the International Traffic in Arms Regulations (ITAR), which, at the time, defined anything cryptographic as an implement of war, requiring an export license. An export license was required not only for physical devices but also for technical data related to them. He claimed our papers constituted such technical data and they were, of course, exported when published internationally.

The IEEE wrote back, telling the NSA employee it was aware of ITAR, but "the burden of obtaining any Government approval for publication of technical data [was] on the person or company seeking publication," namely me and Stanford University.¹⁴ A copy of this reply was sent to me, and I took it to Stanford's General Counsel John Schwartz both because Stanford was potentially liable and because I wanted to ensure it would defend me if I was prosecuted.

Schwartz took a few days to review the matter, after which we had a second meeting. He believed that publishing my papers was lawful but noted there was "at least one contrary view" (expressed by the NSA employee) and "should such view be adopted by the Federal Government you could be subjected to prosecution." He went on to assure me that, should that occur, "the University would defray the reasonable costs of your defense ... nevertheless, there would always remain a risk to you

personally of fine or imprisonment if the government prevailed in such a case.”¹⁹

Schwartz also advised me to change my plans for having two students, Ralph Merkle and Stephen Pohlig, deliver joint papers at the upcoming 1977 IEEE Symposium on Information Theory. He explained that a long court case might kill the career of a newly minted Ph.D., whereas I had tenure. I relayed this to Merkle and Pohlig, telling them I had no qualms about delivering the papers but would leave the decision to them. Both said they would deliver the papers anyway but later changed their minds to assuage fears expressed by their parents.

Wanting these students to get the credit they deserved, when it was time for each paper to be delivered, I had the student co-author stand next to me at the podium. I then told the audience that, on the advice of Stanford’s counsel, I would be delivering the papers, but to give the student the credit he deserved, they should consider the words coming from my mouth as if they were coming from his. This gave Merkle and Pohlig even more credit for their work than if they had delivered the talks without any threats.

Get Curious, Not Furious

This first round of the crypto wars had mixed results. We established that independent researchers could publish papers free of government interfer-

ence, but NSA was able to keep DES’s key size at 56 bits.

Commercial encryption did not become truly secure until some parties on both sides of the battle learned a lesson my wife and I later emphasized in our book⁷—the need to get curious, not furious. Since the emphasis here is cybersecurity, I refer those interested in more personal details to the book’s Chapter 3, also called “Get Curious, Not Furious.” That same shift started a process that led to the strong encryption available on today’s commercial products. It started in 1978 when I received a call from NSA saying its Director, Admiral Bobby Inman, would like to visit me and asking if I was open to the idea.

Up to that point, we had fought these battles indirectly, with no direct interchange, so I jumped at the opportunity. When Admiral Inman came to my office, he told me he was meeting with me against the advice of all the other senior people at the Agency but saw no harm in talking. He was curious, not furious. He also said it was nice to see I did not have horns—which must have been how I was being depicted at the Agency. I returned the compliment, since I had seen myself as Luke Skywalker to NSA’s Darth Vader. I was in my early 30s at the time, so the young-hero model was more appropriate than it would be today, when I am 72 years old. My relationship with Inman was cautious at first but it grew

into friendship as we came to appreciate one another’s concerns.

The real break came in the mid-1990s when Congress requested the National Research Council undertake a study of national cryptographic policy. The study committee represented all major stakeholders, including law enforcement, national security, industry, health care, and privacy. By talking to one another—and, more important, listening to one another—we were able to reach unanimous conclusions that encouraged a significant loosening of the export restrictions on encryption products. This further example of getting curious instead of furious laid the foundation for widespread availability of strong encryption in commercial products, with export restrictions being significantly relaxed soon thereafter.

The value of adversaries talking and listening can also be seen in a 2014 interview with Admiral Inman conducted by Stanford cryptography student Henry Corrigan-Gibbs. When asked if he now would make the same decision he did 40 years ago to try to suppress our work, Inman replied, “Rather than being careful to make sure they [were not] going to damage [NSA’s intelligence operations] . . . I would have been interested in how quickly they were going to be able to make [encryption widely] available.” He cited the theft of portions of the F-35 jet fighter design as proof that strong commercial encryption was in the U.S.’s broader national security interests.²

How Logical Is Cyber-Deterrence?

Nuclear deterrence is viewed so positively that cyber-deterrence is frequently suggested as a promising analogous next step. For example, the current Director of NSA and U.S. Cyber Command, Admiral Michael S. Rogers, told a Senate committee in 2015, “We also need to think about how can we increase our capacity on the offensive side here, to get to that point of deterrence.”¹⁸

But how logical is cyber-deterrence? The answer depends in part on a related question treated in Chapter 8 of our book⁷ (pages 243–264): “How logical is nuclear deterrence?” To summarize, consider these key points:

We must behave irrationally. For deterrence to work in a standoff between



Cryptography pioneers Ralph Merkle, Martin E. Hellman, and Whitfield Diffie, 1977.

the U.S. and another nuclear-armed nation, that adversary must be rational enough to be deterred by our threats, but we must be irrational enough for its equally dire threats not to deter us. This need for irrationality on our part is usually swept under the rug, but a 1995 U.S. Strategic Command report, *Essentials of Post-Cold War Deterrence* (<http://www.nukestrat.com/us/stratcom/SAGessentials.PDF>), was unusually candid. After noting that instilling fear in our adversaries is “the working force of deterrence,” it advised “that the U.S. may become irrational and vindictive if its vital interests are threatened should be part of the national persona we project to all adversaries.”


Nuclear deterrence must be carefully defined. The U.S. has not carefully defined what it means by “nuclear deterrence.” For example, does it mean we have nuclear weapons solely for the purpose of deterring a nuclear attack on us or our allies? That is the impression given by many statements from the U.S. government. But if that is the case, why do we use nuclear threats when the stakes are far lower?

Impaired decision making. World leaders have the power to start a nuclear war even when they cannot legally drive a car. Documented examples of persistent problems with alcohol⁷ (pages 250–251) include Russian President Boris Yeltsin, U.S. President Richard Nixon, and British Prime Minister Tony Blair. I suspect most leaders with “fingers on the button” are occasionally similarly impaired.


Risk. No one knows how risky nuclear deterrence is—a subject discussed in the next section—that then relates the problem to a critical issue in encryption.

Nuclear Deterrence and Cryptography

Surprisingly, there is no evidence that the U.S. government has investigated the risk that nuclear deterrence might fail and thereby destroy civilization. (I strongly suspect the same is true of other nuclear-armed nations but have not investigated them as deeply.) No unclassified information indicates that any such studies exist. While I currently hold no clearances and could therefore be unaware of classified studies, I have discussed the possibility of such studies with sympathetic, high-level people



“... nevertheless, there would always remain a risk to you personally of fine or imprisonment if the government prevailed in such a case.”



with appropriate clearances and no evidence surfaced there either. I also have discussed undertaking such studies with high-level personnel within the U.S. Strategic Command, the successor to the Strategic Air Command, and that, too, did not produce any claims of studies, nor any real interest in investigating the level of risk.

This dearth of information on one of the most important questions facing humanity led me to spend much of the past 10 years working to bring a risk-informed framework to nuclear deterrence. As part of that effort, I published a simplified, preliminary risk analysis⁹ indicating the level of risk is unacceptable.

To put such risk in perspective, even if nuclear deterrence could be expected to work for 500 years before it failed and destroyed civilization—a time horizon that sounds optimistic to most people—it would be equivalent to playing Russian roulette with the life of a child born today. That is because that child’s expected lifetime is roughly one-sixth of 500 years. If the time horizon is more like 100 years, the child’s odds are worse than 50/50.

My work applying risk analysis to nuclear deterrence led me to see an important and largely overlooked question in cryptography. There is much talk today about the need for “post-quantum crypto,” meaning systems that would remain secure even if large quantum computers²⁰ could be built. But there is much less concern about possible advances in algorithms that would render both RSA and the “usual” Diffie-Hellman-Merkle Key Exchange insecure. There should be concern, as we will see. For simplicity in what follows, I talk only about factoring and RSA, but the same arguments apply equally to discrete logarithms and Diffie-Hellman-Merkle Key Exchange.

Factoring algorithms took a major step forward in the 1970s when Morrison and Brillhart¹⁵ used Lehmer’s and Powers’s “continued fraction method”¹¹ to factor the seventh Fermat number, which is 128 bits long.

A second major advance occurred in the 1980s when American mathematician Richard C. Schroepel used sieving to roughly double the size of the numbers that could be factored. He never published his algorithm but cir-


culated it to many relevant researchers, and Carl Pomerance credits Shroepel's algorithm as "the forerunner of [Pomerance's better known] quadratic sieve and also its inspiration."¹⁶

A third major advance in factoring occurred in the 1990s with development of the "number field sieve," again roughly doubling the size of numbers that could be factored.


While major advances in factoring occurred in the 1970s, 1980s, and 1990s, no similar advances have occurred in roughly the past 25 years, leading many mathematicians and cryptographers to believe that factoring has hit a brick wall. But I see the situation quite differently as a result of my work applying risk analysis to nuclear deterrence.¹⁰

Think of each decade as a coin toss that shows heads if a major advance occurs in factoring and tails otherwise. The 1970s gave us heads, as did the 1980s and 1990s, but the next decade gave us tails, and the current decade is more than half over without a major advance, so it seems more likely than not to also give us tails. Even under the optimistic assumption that no major advance occurs in the remaining years of this decade, the coin-toss sequence would be HHHTT. If a coin showed such a sequence in its first five tosses, it would be foolish to project tails into even the next decade of the 2020s with any reasonable degree of confidence.

Given the impact another major advance in factoring would have on the global economy, I have argued that it would be prudent to already have backup systems for both key exchange and digital signatures in place and in use. For key exchange, two keys could be generated and hashed or XORed. One key would be produced by public key exchange and the other by the backup system. Such a system would provide seamless security even if one of the methods of key exchange were compromised. One possible backup system would be a key distribution center that shares a master key with each user and distributes session keys on demand, encrypting the session key in each relevant user's master key. Likewise, two digital signatures could be used to sign each message, with a possible backup system being Merkle's tree signatures.¹³



Logic is just one way of knowing about the world, and an incomplete one at that.



Alan Turing and My Illogical Use of Logic

The section "Illogical Logic" in our book⁷ (pages 244–251) describes how supposedly highly logical people can misuse logic. In keeping with the book's aim to move from blame to responsibility, the first story in the section describes how, years ago, I misused logic as a weapon to win arguments with my wife. While I may have been winning arguments (at least in my mind), I was losing something much more important—my relationship with her. Illogical logic loses every time.

That section also describes how I felt like I was having a mental breakdown when confronted with Gödel's Incompleteness Theorem in my second year of graduate studies at Stanford. I had based my whole life on logic—not just my professional life—and logic was telling me it was literally incomplete. Because it would have complicated matters too much for the average reader, we purposely left out of that section Alan Turing's role in creating my angst. But my ACM Turing Lecture provided a wonderful opportunity to highlight how Turing helped open my mind to new possibilities.

In that second-year graduate math course, we studied the cardinality of infinite sets. The positive integers are "countably infinite" because you can count or enumerate them 1, 2, 3, . . . It was easy to see that the set of all integers is also countably infinite, with one enumeration being 0, -1, +1, -2, +2, and so on. Every integer is eventually reached in that enumeration.

It was slightly more difficult to see that the set of rational numbers is countably infinite. For simplicity, I show only the argument for positive rational numbers, though it extends easily to all rational numbers. The countably infinite sequence $1/1$; $1/2$, $2/1$; $1/3$, $2/2$, $3/1$; $1/4$, $2/3$, and so on includes all positive rational numbers. (I use semicolons to demark the end of subsequences in which numerators and denominators have a common sum, as in $1/3$, $2/2$, and $3/1$.)

Things became much more interesting when the professor showed that the real numbers were "uncountably infinite"; that is, they form a larger infinite set that cannot be enumerated. The proof was by contradiction, using Georg

Cantor’s “diagonalization argument.” Assume there is an enumeration of the real numbers

$$\begin{aligned} R_1 &= I_1 . b_{11} b_{12} b_{13} \dots \\ R_2 &= I_2 . b_{21} b_{22} b_{23} \dots \\ R_3 &= I_3 . b_{31} b_{32} b_{33} \dots \text{ and so on} \end{aligned}$$

where R_i is the (assumed) i^{th} real number, I_i is its integer part, and b_{ij} is its j^{th} binary decimal place. To complete the proof, consider the real number

$$R = 0 . \sim b_{11} \sim b_{22} \sim b_{33} \dots$$

where $\sim b_{ij}$ is the complement of b_{ij} . This real number R is different from R_1 since they differ at least in their first binary decimal places. It is different from R_2 , since they differ at least in their second binary decimal places. Similar arguments apply to each R_i in the assumed list. We had assumed the list included all the reals, but R is not in the enumeration, so the reals are not countably infinite.

So far, I was not too perturbed. But then the professor defined the computable real numbers, a concept first introduced by Turing in his brilliant 1936 paper.²¹ A computable real number is one that can be computed to as many decimal places as desired in a finite (though indeterminate) time by a finite-length program. While, at first, this set might seem to depend on the machine being used, that problem was removed by using a Universal Turing Machine that can simulate any other physical computer with only a finite increase in program size and run time over what would be needed on the machine being simulated.

The set of finite-length programs can clearly be enumerated, as in 0, 1, 00, 01, 10, 11, 000, and so on. Since not every finite-length program produces a computable real number—some get hung up in infinite loops and provide no output—the set of computable real numbers is also countably infinite. But the professor then seemed to prove that the computable real numbers were uncountably infinite by writing the following program:

Print 0 and a (binary) decimal point, so that what follows is the binary expansion of a computable real number.

```
FOR i=1, i++
{Compute bii the ith binary
decimal place of the ith
computable real number, and
print ~bii}
```

This reasoning, drawn from Section 8 of Turing’s paper, is almost exactly the same as was used to prove the real numbers are not countable. But there is a difference, as there must be, since the “proof” here produced a contradiction to a known fact: The computable real numbers are countable.

This line of reasoning involves a very subtle, hidden assumption—that there exists a computable enumeration of the computable real numbers. An enumeration exists, but we can never compute it. In a sense, only God knows it, while we mortal humans cannot.

I was dumbfounded. If an incorrect assumption can be that subtle, what others might have been missed in other proofs? Is mathematics itself on a firm foundation? Might Cantor’s proof that the reals are uncountably infinite have a similar flaw? (I still wonder about that.)

My world was shaken in that course, but not enough for me to give up logic as the primary basis for my personal and professional life. That took 10 more years and almost ruining my marriage before I finally accepted what Gödel and Turing had been implicitly telling me: Logic is just one way of knowing about the world, and an incomplete one at that.

I learned the limits of logic in time to save my marriage. Will humanity learn the limits of its current logic in time to save the world and itself? Dorothea and I wrote our book partly to increase those odds, even if just a bit. That provides yet one more connection between the work that won me the ACM A.M. Turing Award and the book. What is the point of developing elegant algorithms (such as Diffie-Hellman-Merkle Key Exchange) if no one is around in 100 years to use them? □

References

1. Coppersmith, D. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development* 38, 3 (May 1994), 243–250.
2. Corrigan-Gibbs, H. Keeping secrets. *Stanford Magazine* (Nov./Dec. 2014), 58–64; <http://tinyurl.com/cyptowar1>, expanding to https://alumni.stanford.edu/get/page/magazine/article/?article_id=74801
3. Diffie, W. and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*

- 10, 6 (June 1977), 74–84; <http://www-ee.stanford.edu/%Ehellman/publications/27.pdf>
4. Diffie, W. and Hellman, M. New directions in cryptography. *IEEE Transactions on Information Theory IT-22*, 6 (Nov. 1976), 644–654; <http://www-ee.stanford.edu/%Ehellman/publications/24.pdf>
5. Ellis, J. The history of non-secret encryption. *Cryptologia* 23, 3 (1999), 267–273.
6. Graham, S. Letter to Ralph C. Merkle (Oct. 22, 1975); <http://www.merkle.com/1974/RejectionLetter.pdf>
7. Hellman, D. and Hellman, M. *A New Map for Relationships: Creating True Love at Home & Peace on the Planet*. New Map Publishing, Stanford, CA, 2016; <http://tinyurl.com/HellmanBook>, expanding to <http://www-ee.stanford.edu/%Ehellman/publications/book3.pdf>
8. Hellman, M., Merkle, R., Schroepel, R., Washington, L., Diffie, W., Pohl, S., and Schweitzer, P. Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard. Technical Report SEL 76-042 (available from NTIS), Electrical Engineering Department, Stanford University, Stanford, CA, Sept. 9, 1976 (revised Nov. 10, 1976); <http://tinyurl.com/nbs-des-analysis>, expanding to http://www-ee.stanford.edu/~hellman/resources/1976_sel_des_report.pdf
9. Hellman, M. Risk analysis of nuclear deterrence. *The Bent of Tau Beta Pi* 99, 2 (Spring 2008), 14–22; <http://tinyurl.com/hellman74>, expanding to <http://www-ee.stanford.edu/%Ehellman/publications/74.pdf>
10. Hellman, M. How risky is nuclear optimism? *Bulletin of the Atomic Scientists* 67, 2 (Mar. 2011), 47–56; <http://tinyurl.com/HowRisky>, expanding to <http://www-ee.stanford.edu/~hellman/publications/75.pdf>
11. Lehmer, D. and Powers, R. On factoring large numbers. *Bulletin of the American Mathematical Society* 37, 10 (1931), 770–776.
12. Merkle, R. Secure communication over insecure channels. *Commun.* 21, 4 (Apr. 1978), 294–299.
13. Merkle, R. A digital signature based on a conventional encryption function. In *Advances in Cryptology, CRYPTO 1987, Lecture Notes in Computer Science, Vol. 293* (Santa Barbara, CA, Aug. 16–20). Springer-Verlag, Berlin, Heidelberg, Germany, 1988, 369–378.
14. Meyer, J. Letter to IEEE (July 7, 1977); <https://stacks.stanford.edu/file/druid:wg115cn5068/1977%20070%20Meyer%20letter.pdf> and <https://purl.stanford.edu/wg115cn5068>
15. Morrison, M. and Brillhart, J. A method of factoring and the factorization of F7. *Mathematics of Computation* 29, 129 (Jan. 1975), 183–205.
16. Pomerance, C. A tale of two sieves. *Notices of the AMS* 43, 12 (Dec. 1996), 1473–1485.
17. Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (Feb. 1978), 120–126.
18. Sanger, D. U.S. must step up capacity for cyberattacks, chief argues. *The New York Times* (Mar. 20, 2015), A4.
19. Schwartz, J. Memo to Martin Hellman (Oct. 7, 1977); <https://stacks.stanford.edu/file/druid:wg115cn5068/1977%201007%20Schwartz2MH.pdf>
20. Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26, 5 (1997), 1484–1509.
21. Turing, A. On computable real numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2* 42 (1936), 230–265; https://www.cs.virginia.edu/~robin/Turing_Paper_1936.pdf

Martin E. Hellman (martydevoe@gmail.com) is Professor Emeritus of Electrical Engineering at Stanford University, Stanford, CA.

Copyright held by author.
Publication rights licensed to ACM, \$15.00



Watch the author discuss his work in this exclusive *Communications* video. <https://cacm.acm.org/videos/cybersecurity-nuclear-security-alan-turing-and-illogical-logic>

DOI:10.1145/3152422

Even when surrounded by ubiquitous computing, humans should be encouraged to do what they do better than machines.

BY RAMIRO MONTEALEGRE AND WAYNE F. CASCIO

Technology-Driven Changes in Work and Employment

WE LIVE IN a global society where technology, especially information and communication technology, is changing the way businesses create and capture value, how and where we work, and how we interact and communicate. In her seminal 1988 book, *In the Age of the Smart Machine: The Future of Work and Power*,⁴⁵ Shoshana Zuboff was among the first scholars to weave together the technological, sociological, and psychological processes that have converged to shape the modern workplace. Her insights concerned the nature of information and its significance in restructuring and redefining the patterns and meanings of work, even though at the time of her study the worldwide diffusion of the

Internet had not yet occurred. Academic literature, not only in business^{9,32,42} but also in medicine,^{15,38} engineering,^{23,40} physical sciences,³⁰ and social sciences,^{21,37} echo these observations in more recent times. To illustrate the effects of the changes on organizations, we consider their implications for the management of human talent.

The new wave of technological innovation features the emerging general paradigm known as “ubiquitous computing,”^a or an environment where computational technology permeates almost everything, enabling new ways of connecting people, computers, and objects. Ever-cheaper cost for computation has resulted in the proliferation of computing devices, including personal computers, embedded (enabled by micro-miniaturization) and networked industrial sensors and processors, speech-recognition and eye-tracking devices, mobile devices, radio-frequency-identification and near-frequency-communication tags and labels, global-positioning-systems-enabled devices, smart televisions, car navigation systems, drones, wearable sensors, robots, and 3D virtual reality. The ubiquitous computing infrastructure also enables collection

a The term “ubiquitous computing” was coined by Mark Weiser in 1988 at Xerox Palo Alto Research Center; in Latin, “ubiquitas” means being everywhere. See also *Communications* February 2002 special section on ubiquitous computing.²⁵

» key insights

- **As technology keeps advancing, we need to think beyond augmenting or automating jobs to how to manage the messy process of the creative destruction of jobs as we create the new ones.**
- **What enables or constrains people in the workplace is the way they use and manage technology, not technology itself.**
- **Technology-driven changes demand from us an understanding of the technology in relation to the entire work system, the relational and non-relational roles and interactions of human participants and/or machines.**



IMAGE BY ZENZEN

of enormous quantities of structured and unstructured data, requiring the adjective “big” to distinguish this new paradigm of development. Ubiquitous computing also blurs the boundaries between industries, nations, companies, providers, partners, competitors, employees, freelancers, out-sourcers, volunteers, and customers. They also yield opportunities to unify the physical space, which has always used information to try to make an inherently inefficient system more efficient, and the electronic space, which enables information accessibility to

overcome the limitations of the physical space. Merging the physical and the electronic also has implications for privacy and security, as well as how companies are organized and manage human talent.

Given these rapid advances and our increased reliance on technology, the question of how to manage technology-enabled change in work and employment is highly salient for companies and their executives. General predictions anticipate significant changes in knowledge acquisition, sharing, and distribution, as well as related ripple

effects in the workplace.^b Work is defined here as the application of human, informational, physical, and other resources to produce products and services.⁵ If one accepts that work does not exist without people and executives are inherently concerned with the management of people within organizations, then they bear some responsi-

^b See, for example, Marc Andreessen’s *Wall Street Journal* column “Why Software Is Eating the World” (Aug. 20, 2011); <http://www.wsj.com/articles/SB10001424053111903480904576512250915629460>

bility for understanding the effects of technology on work and employment. This article thus aims to interpret the progress, direction, and managerial implications of current research in work and employment. We begin with three lessons for executives based on our review of relevant literature. We then examine how technology affects six key areas of talent management as organizations move from traditional to ubiquitous computing. We conclude with a series of questions for managers in the six areas.

Methodology

This article is part of a larger project aimed at examining how technology is changing work and organizations.¹² Our conclusions are based on a comprehensive review of the literature in management, industrial/organizational psychology, labor economics, human-factors design, and information and computer technology.

Lesson 1. The effect of ubiquitous computing on jobs is a process of creative destruction. Ubiquitous computing is not the first technology to affect jobs. From steam engines to robotic welders to ATMs, technology has long displaced human workers, often creating new and higher-skilled jobs in its wake. Mass production of the automobile threw many blacksmiths out of work but also created far more jobs building and selling cars. Over the past 30 years, the digital revolution, coupled with global business markets, have displaced many of the middle-skill jobs behind 20th century middle-class life in Western industrial countries. The number of typists, cashiers, travel agents, bank tellers, and production-line jobs has fallen dramatically, particularly in the U.S. and Europe, but there are more computer programmers and web designers than ever before. Displaced workers with obsolete skills are always hurt, but the total number of jobs has never declined over time.²

Paradoxically, although productivity, a key indicator of growth and wealth creation, is at record levels and technological innovation has never been greater, over the past several decades, median wages in the U.S. have not risen.¹⁹ This pattern is inconsistent with economic theory, which holds that when pro-

ductivity increases, any automation that reduces the need for labor will increase business revenue and personal income. That will, in turn, generate demand for new products and services, that will, likewise, create new jobs for displaced workers. One explanation for this pattern is that advances in information and communications technology are destroying more jobs in developed economies than the advances are creating. Technological progress is thus eliminating the need for many types of jobs, leaving the typical worker worse off than before.¹⁰ According to one 2017 study,¹⁸ approximately 47% of total U.S. employment is at risk of automation.

Not all researchers concur with this conclusion, however. Although labor economists generally agree that the digital revolution is opening a great divide between a skilled and wealthy few and everyone else, hollowing out the middle class,⁷ it is not clear that all of it can be attributed to the effects of technology. The data is far from conclusive. One result of the change is the simultaneous increase in both job openings and unemployment relative to the early 2000s,¹⁷ suggesting the types of skills in demand by employers today do not match up with those of the existing labor force. Other plausible explanations, including events related to global trade and the financial crises of the early and late 2000s, could account for the relative slow pace of job creation since the turn of the century. The problem for researchers and executives is that it is difficult to separate the effects of technology from other macroeconomic effects.³⁹

To be sure, the advent of machine learning, where computers teach themselves tasks and rules by analyzing large datasets, will lead to large-scale worker dislocation, as automated areas (such as speech recognition, pattern recognition, and image classification) eliminate large numbers of white-collar jobs.¹⁸ We agree that many jobs performed by humans today, notably bookkeepers, auditing clerks, financial analysts, graphic designers, and medical transcribers, will be substantially taken over by robots or digital agents by 2025. Other jobs will disappear as a result of structural changes in the economy (such as the long-term decline in demand for coal, as cleaner

sources of energy become cheaper and more readily available).

Unlike effective managers, however, machines have not yet learned to tolerate high levels of ambiguity or to inspire people at every level in organizations. Consider ambiguity. The bigger and broader the question to be addressed, the more likely human synthesis will be required to address it because, although machines can provide many pieces of the solution, they cannot assemble the “big picture.” The process of assembly entails discerning why a company is doing what it is doing, where it is trying to go, and how it proposes to get there. Success depends on the ability of executives to tolerate ambiguity and synthesize and integrate a variety of types and forms of information. The big picture represents the glue that holds a company together. Moreover, when it comes to engaging and inspiring people to move in the same direction, empathizing with customers, and developing talent, humans will continue to enjoy a strong comparative advantage over machines.

Even if today’s information and communication technologies limit the potential growth of employment, history suggests it is a temporary, though painful, shock. As workers adjust their skills and entrepreneurs create opportunities based on the new technologies, the number of jobs will rebound. At the same time, human ingenuity will create new jobs, industries, and ways to make a living, just as it has since the dawn of the Industrial Revolution,⁴¹ following Joseph Schumpeter’s “gale of creative destruction.”


Lesson 2. Ubiquitous computing can be used to enable or constrain people at work. To illustrate how that works, consider electronic monitoring systems, robots, and wearable computing devices. Each shares computer science’s expressed ubiquitous computing vision of interweaving technology into everyday life, making technology pervasive, and facilitating physical and virtual interactions.

Electronic monitoring systems. Monitoring refers to systems, people, and processes used to collect, store, analyze, and report the actions or performance of individuals or groups on the job.^{3,8} Our focus here is on electronic monitoring and surveillance systems.


Monitoring today may assume a variety of forms (such as telephone, video, Internet, and Global Positioning Systems). In the past, U.S. courts generally sided with employers when choosing to monitor their employees, arguing that because monitoring takes place during work hours through organizational assets (such as corporate computer networks and email systems), monitoring is acceptable.²²

Many organizations equip machinery, shipments, infrastructure, devices, and even employees with networked sensors and actuators that enable them to monitor their environment, report their status, receive instructions, and take action based on the information they receive. By monitoring these resources in real time, companies can better control the flow of operations and avoid disruptions by taking immediate action as problems arise. Organizations are also developing policies on using blogs and social networks (such as Facebook) outside of work, potentially affecting employees' perceptions of trust and loss of personal control.²⁶ Monitoring per se is neither good nor bad, depending instead on how it is implemented. To be sure, monitoring can be beneficial, as self-initiated systems demonstrate. Systems that enable employees to track their activities at work have led to increased productivity by helping them understand better how they allocate their time.³³ Such understanding allows workers to reallocate their time, tasks, and activities to accomplish work goals more effectively.

A comprehensive review of research in this area concluded that attitudes in general, and attitudes toward monitoring in particular, will be more positive when organizations monitor their employees within supportive organizational cultures.⁴ Supportive cultures welcome employee input into the monitoring system's design, focusing on groups of employees rather than singling out individuals, and focusing on performance-related activities. Theoretical and empirical researchers have identified three additional features of monitoring systems that contribute to employee perceptions of fairness or invasiveness:⁶ consistency in how data is collected and used; freedom from bias (such as selective monitoring); and the accuracy of the data being collected.



Technological progress is eliminating the need for many types of jobs, leaving the typical worker worse off than before.



Conversely, when monitoring systems are viewed as invasive or unfair, organizations run the risk that employees may not comply with rules and procedures, slack off on the job, or engage in deviant behavior.⁴

It is important to note an additional factor that may be associated with electronic monitoring systems—when organizations impose control they reduce autonomy and increase perceived job demands, both contributing to employee burnout.³¹ Evidence from a variety of manufacturing contexts indicates that close supervision is associated with increased stress.²⁴ With electronic monitoring, a supervisor or higher-level manager need not even be present to do the monitoring. As a result, the potential for constant monitoring creates a type of control employees often regard as particularly stressful. As a general conclusion, when electronic monitoring is seen as control-based rather than developmental, employees are likely to experience more negative outcomes.¹³

Robots. Robots^c have been on factory floors for decades. Years ago, they were mostly big, expensive machines that had to be surrounded by cages to keep them from smashing into humans. They could typically perform only a single task (such as spot welding) over and over, albeit extremely quickly and precisely. They were neither affordable nor practical for small businesses. Today, however, so-called collaborative machines are designed to work alongside people in close settings. They cost as little as \$20,000 and offer small businesses incentives to automate in order to increase overall productivity and lower labor costs.¹ Moreover, advances in artificial intelligence, combined with improved sensors, are making it possible for robots to make more complex judgments and to learn to execute tasks on their own, enabling them to manage well in uncertain and fluid situations, many involving humans.


Not only are robots being embedded into organizational social systems, they are becoming social actors within

^c *The Oxford Dictionary* defines “robot” as “A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer”; http://www.oxforddictionaries.com/us/definition/american_english/robot


those systems. Historically, the terms “co-worker” and “teammate” implied fellow humans, but this may no longer be the case, as co-worker robots, or “co-bots,” enter the workplace as team members.¹⁴ As they evolve, robots are likely to become more adaptable to the work environment, with multimodal interfaces enabling them to communicate more efficiently and effectively with human teammates, receiving, as well as transmitting, information.³⁶

A key challenge to human-factors specialists is how to design human-robot control interfaces that are simple and easy to use yet robust, because the connections that allow remote robots to take action without a human operator could be subject to hacking. Social acceptance is critical. If robots are truly to be team members, humans must accept them, communicate effectively with them, develop shared mental models with them, and perhaps, most important, trust them. As robots perform more and more autonomous tasks, operators’ workloads should, in theory, decrease, freeing them to perform other tasks. Yet the allocation of functions between humans and robots is an area that needs considerable attention because automation has been shown to create its own set of problems, including decreased situational awareness, distrust of automation, misuse, abuse, and disuse, complacency, decrements in vigilance, and negative effects on other facets of human performance.³⁶ Research and theory in work analysis, teams, selection, training, motivation, and performance management can aid successful design and integration of robots into work teams and organizations.^{14,28}

There is an additional concern that managers must address—that workers view robots as competitors for jobs and resist their installation. For surviving workers, robots can indeed augment their capabilities, but the fear of job loss is real. At Fanuc Corporation’s 86,000-square-foot factory in Oshino, Japan, which makes industrial robots, only four people staff the entire factory. In another, robots can assemble an industrial motor in just 40 seconds.³⁴ Robots threaten the jobs of white-collar workers as well. As an example, consider that robots now perform work in corporate finance departments that used



The potential for constant monitoring creates a type of control employees often regard as particularly stressful.



to require teams of people, as software automates many corporate bookkeeping and accounting tasks. Between 2004 and 2015, the median number of full-time employees in the finance department at big companies declined 40%, from 119 to approximately 71 people for every \$1 billion of revenue.²⁹ Jobs most in jeopardy include accounts-payable clerks, inventory-control analysts, and accounts-receivable clerks who send invoices to customers, track payments, and forecast customer default rates.²⁹

Not all robots or robot makers will displace humans, however. For example, Kiva robots, owned and manufactured by Amazon Robotics, is designed to scurry across large warehouses, fetching racks of ordered goods and delivering them to humans, who package the orders. A warehouse equipped with Kiva robots can handle up to four times as many orders as a similar unautomated warehouse, where human workers might spend as much as 70% of their time walking or transporting themselves to retrieve ordered goods. Most of Kiva’s customers are e-commerce retailers, some growing so quickly they cannot hire people fast enough. By making distribution operations cheaper and more efficient, robotic technology has helped many of these retailers survive and even expand. Such advances illustrate that while some aspects of work can be automated, humans still excel at certain tasks (such as packaging various items together). Kiva robots are designed and built to work with people, taking over tasks humans do not want to do or are not very good at. While they can enhance the productivity of these workers, clerical and some professional jobs could be more vulnerable, as the marriage of artificial intelligence and big data gives machines more human-like abilities to reason and solve new types of problems.³⁹

Wearable computing devices, or “wearables.” Wearables^d generally comprise three broad categories:⁴⁴ “quantified self” products that allow people to measure their activities (such

^d *The Oxford Dictionary* defines “wearable” as “Denoting or relating to a computer or other electronic device that is small or light enough to be worn or carried on one’s body”; http://www.oxforddictionaries.com/us/definition/american_english/wearable

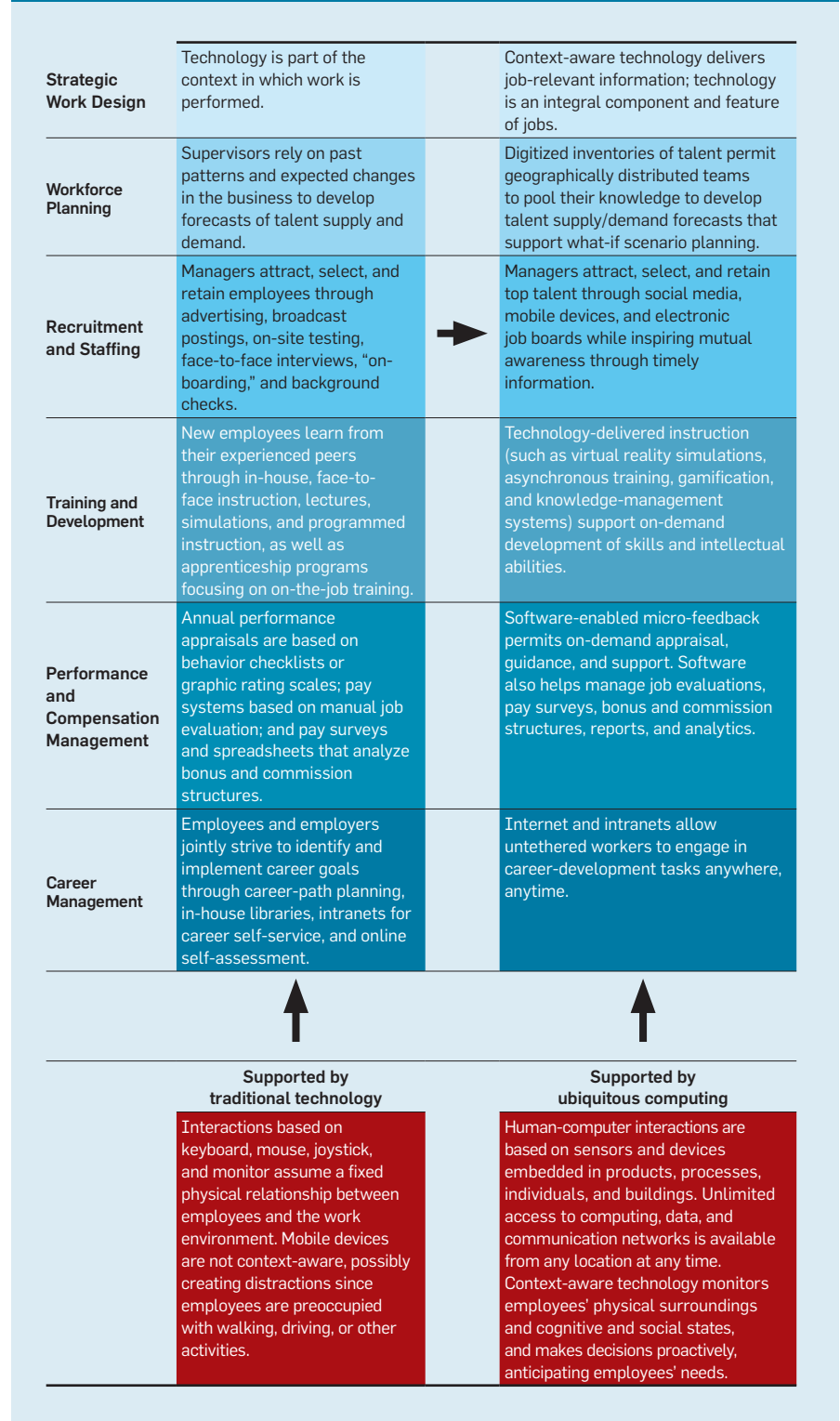
as physical activity and sleep, as with Fitbit and Jawbone); enhancement technologies (such as Google Glass, prosthetic devices, and exoskeletons that help elderly people or those with disabilities); and virtual reality devices (such as headsets and telepresence systems), as with architects using them to see what their designs will look like in practice. Telepresence systems enable executives to experience the feeling of “being there,” attending meetings without having to travel. These devices are now possible thanks to four developments: improved computing power, increased speed of broadband access, the spread of sensors, and cloud computing.⁴⁴ Smart vending machines are another example of how the nature of work is changing. Embedded sensors, combined with broadband access and cloud computing, make it possible to monitor them remotely for items out of stock, temperature changes, and pilferage. While the promise of wearable computing devices is obvious, there are potential drawbacks as well. The first is distraction, as people are cognitively half present and half absent, constantly checking their smartphones as they walk along or stand in line. How often? They check them an average of 3.1 hours a day, according to one study by Meeker.²⁷ This can wreak havoc on work/life integration, as there is no boundary by time or geography as to when or where people work.⁴³ Another drawback is that digital devices make human interaction more difficult as the devices compete constantly for people’s attention.

Despite the drawbacks, many uses of wearable technology are emerging beyond consumer applications, becoming popular in industries as varied as construction, building maintenance, medicine, manufacturing, energy, and oilfield services. As an example, consider how a company in building maintenance might use wearables to preserve and transmit institutional memory. Workers nearing retirement are not always well-suited to climbing ladders or scaffolding to significant heights where mechanical equipment might be located. They leave that task to younger workers wearing special safety glasses equipped with cameras, microphones, speakers, detachable flash drives, and wireless antennas.

Through Bluetooth connections to their phones, the younger workers could then transmit live video feeds of their actions back to a ground-based command center staffed by veteran older workers monitoring the videos and offering further guidance.²⁰

Lesson 3. Ubiquitous computing is changing the nature of competition, work, and employment in ways that are profound and that need active management. Before it was possible to access inexpensive computational support, hoarding information was a source of

Figure 1. Six areas of talent management supported by traditional and ubiquitous computing technologies.



power, and information moved in one direction only—up the corporate hierarchy. In today’s business environment of ubiquitous computing, the contrast could not be starker. While the changes made possible by today’s technology might be impressive, and digital innovation will continue for the foreseeable future, technology by itself does not ensure profitable business performance.

A comprehensive 2014 review of research at the junction of leadership and technology concluded that researchers tend to treat technology either as a contextual aspect of business performance relevant to the leadership process or as a set of tools that leaders and followers can use to communicate with each other.³⁵ The complex, pliable, changing, and ever-expanding portfolio of Internet tools, information, and media is altering how consumers and businesses act in situations where previously they would have acted differently. Before the Internet, it was impossible to,

say, communicate instantaneously or asynchronously across time and space or access vast bodies of information without visiting a library or other physical repository.

With the Internet, people have easy access to information they previously could not have found. Indeed, technologies trigger change by altering workers’ non-relational roles—the business-related tasks they perform and how they perform them. These changes may then lead to changes in the nature of the interactions workers have with other members of their role set, or fellow workers with whom they interact while doing their work, as well as others in their role set (such as co-bots). If role relations change in either way, then the social network is likely to change as well. If it does, one can say technology has altered the work system. Changes in role relations are thus key to a broad range of effects in work systems. To be sure, technology is altering role relations in profound ways.

Technology and Talent Management

The way technology is altering work settings and the work people do, particularly in the new era of ubiquitous computing, affects the way organizations manage their human talent and raises compelling questions for managers. Consider pre-employment testing. Traditionally, candidates would take tests at an employer’s site, in a quiet, distraction-free, comfortable place, where the employer could prevent breaches of security by checking candidate identification, eliminating opportunities for collusion, and controlling test materials at all times. Now consider unproctored Internet testing, where candidates, not employers, decide what conditions are best. Technology can deliver simulations or pre-employment assessments to any location at any time, raising a number of other security and trust issues that might influence test outcomes of interest, including the reliability and validity of the measures, adverse impact, size of the applicant pool, differences in means and standard deviations, applicant reactions, and perceptions of procedural justice.

There is certainly great potential for deepening management’s understanding of and ability to predict behavior in the domain of technology and talent management. Figure 1 outlines how the shift from traditional to ubiquitous computing technologies affects six conventional areas of talent management:¹¹ work design, workforce planning, recruitment and staffing, training and development, performance management and compensation management, and the management of careers. Figure 2 outlines key questions for managers when moving from traditional to ubiquitous computing technology in these areas. Note the relevance of the lessons mentioned earlier, particularly lesson 2—that ubiquitous computing can be used to enable or to constrain people at work—as managers seek to address the questions in Figure 2.


Conclusion

Research on technology and organizations provides valuable insight re-

Figure 2. Questions for managers when moving from traditional to ubiquitous computing in six areas of talent management.

| | |
|--|---|
| Work Design | How does unlimited access to computer-based resources change communication, document sharing, knowledge exchange, and collaboration in work settings? How can technology enable job design that advances, rather than threatens, innovation, fulfilling work, and value creation? How might the design of work reduce stress associated with constant connectivity? |
| Strategic Workforce Planning | What are the desired and unintended effects of the increased ability to receive and process rich streams of data about the organization and its environment? How does ubiquitous computing affect workforce collaboration, cohesion, and performance? How might technology and ubiquitous computing help minimize risk in workforce supply-and-demand forecasts? |
| Recruitment and Staffing | Given the volume of digitized data, what legal, ethical, privacy, and fairness issues are associated with screening and tracking individuals in and outside an organization? How is the role of the recruiter changing in a world of constant connectivity? What effect does technology-based staffing have on productivity at the individual and the enterprise level? |
| Training and Development | How can technology-delivered instruction enhance employee and team training? Just as there are “smart cars” and “smart buildings,” how can organizations enable and support “smart workers”? How can new training technologies like virtual reality, e-learning, and gamification enhance training outcomes? |
| Performance and Compensation Management | What strategies promote sensible performance management and fair compensation in digital work environments? How do social ties and non-work-related communication affect performance in a world of unlimited connectivity? What are the most effective ways to supervise employees in ubiquitous-technology work environments? |
| Career Management | What are the best ways to coach employees to self-manage their careers? What kinds of technology could enhance this process? How might technology facilitate work/life fit? What roles do personal control, collaboration, and coordination of career management play in the digital environment? |

garding what managers know about the effects of technology. Based on a review of this research, we identified three main conclusions about how ubiquitous computing affects work and organizations: how the effect on jobs reflects a process of creative destruction; how it can be used to enable or constrain people at work; and how it is changing the nature of competition, work, and employment in ways that are profound and that need to be actively managed. We explored the effects of ubiquitous computing on six key areas of talent management, identifying a series of questions to help guide decision making as managers transition from traditional to ubiquitous computing in these areas.

Ultimately, the critical issue for managers to consider is not technology itself but that technology is fundamentally social, grounded in specific historical and cultural contexts. As it becomes embedded in everyday activities and social relations, technology affects all manner of human and organizational elements (such as governance structures, work routines, information flow, decision making, human interactions, and social actions). Fulfilling the potential of technology in work and employment will thus require recreating the way organizations operate in a world of digital ubiquity to maximize positive consequences for individuals and organizations and minimize the negative. Managing in a manner that inspires human performance includes framing the right questions, responding to exceptional circumstances highlighted by intelligent algorithms, and letting humans do things machines cannot.¹⁶ Each organization's leaders, along with other stakeholders, must decide what technologies are adopted, how they are implemented, and the extent to which they augment or detract from worker autonomy, personal competence and control, and interpersonal connections with other human workers. At a broader level, there is a strong need for responsible public policies across institutions, not only to enhance competition, maximize economic surplus, and optimize its allocation across stakeholders, but also to minimize social and human risks and abuses. Establishing such policies will be an ongoing challenge for years to come. 

References

- Aepfel, T. Robots work their way into small factories. *The Wall Street Journal* (Sept. 18, 2014), B1–B2; <https://www.wsj.com/articles/robots-work-their-way-into-small-factories-1410979100>
- Aepfel, T. What clever robots mean for jobs. *The Wall Street Journal* (Sept. 24, 2015); <http://www.wsj.com/articles/what-clever-robots-mean-for-jobs-1424835002>
- Alge, B.J. Effects of computer surveillance on perceptions of privacy and procedural justice. *Journal of Applied Psychology* 86, 4 (Aug. 2001), 797–804.
- Alge, B.J. and Hansen, S.D. Workplace monitoring and surveillance research since 1984: A review and agenda. In *The Psychology of Workplace Technology*, M.D. Covert and L.F. Thompson, Eds. Routledge, New York, 2014, 209–237.
- Alter, S. Work system theory: Overview of core concepts, extensions, and challenges for the future. *Journal of the Association for Information Systems* 14, 2 (Feb. 2013), 72–121.
- Ambrose, M.L. and Alder, G.S. Designing, implementing, and utilizing computerized performance monitoring: Enhancing organizational justice. In *Research in Personnel and Human Resources Management*, G.R. Ferris, Ed. JAI Press, Greenwich, CT, 2000, 187–219.
- Autor, D.H. and Dorn, D. The growth of low-skill service jobs and the polarization of the U.S. labor market. *American Economic Review* 103, 5 (2013), 1533–1597.
- Ball, K. Workplace surveillance: An overview. *Labor History* 51, 1 (Apr. 2010), 87–106.
- Barley, S.R. Why the Internet makes buying a car less loathsome: How technologies change role relations. *Academy of Management Discoveries* 1, 1 (June 2015), 31–60.
- Brynjolfsson, E. and McAfee, D. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W.W. Norton, New York, 2014.
- Cascio, W.F. and Aguinis, H. Research in industrial and organizational psychology from 1963 to 2007: Changes, choices, and trends. *Journal of Applied Psychology* 93, 5 (2008), 1062–1081.
- Cascio, W.F. and Montealegre, R. How technology is changing work and organizations. *Annual Review of Organizational Psychology and Organizational Behavior* 3, 6 (2016), 349–375.
- Castanheira, F. and Chambel, M.J. Reducing burnout in call centers through HR practices. *Human Resource Management* 49, 6 (Nov./Dec. 2010), 1047–1065.
- Covert, M.D. and Thompson, L.F., Eds. *The Psychology of Workplace Technology*. Routledge, New York, 2014.
- Demaerschalk, B.M., Vargas, J.E., Channer, D.D., Noble, B.N., Kiernan, T.J., Gleason, E.A., Vargas, B.B., Ingall, T.J., Aguilar, M.I., Dodick, D.W., and Bobrow, B.J. Smartphone teleradiology application is successfully incorporated into a telestroke network environment. *Stroke* 43, 11 (Sept. 2012), 3098–3101.
- Dewhurst, M., and Willmott, P. Manager and machine: The new leadership equation. *McKinsey Quarterly* (Sept. 2014); http://www.mckinsey.com/insights/leading_in_the_21st_century/manager_and_machine
- Elsby, M., Hobijn, B., and Sahin, A. The labor market in the Great Recession. In *Brookings Papers on Economic Activity, Spring 2010*, D. Romer and J. Wolfers, Eds. The Brookings Institution, Washington, D.C., 2010.
- Frey, C.B. and Osborne, M.A. The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change* 114 (Jan. 2017), 254–280.
- Galston, W.A. Countering tech's damaging effect on jobs. *The Wall Street Journal* (Oct. 14, 2014); <http://www.wsj.com/articles/william-galston-countering-techs-damaging-effect-on-jobs-1413328435>
- Griffith, E. Wearable technology. *Fortune* (Oct. 27, 2014), 57–60; <http://2v1p011c9d9y1a3zveg9dmu6.wpengine.netdna-cdn.com/wp-content/uploads/2015/04/84563.pdf>
- Hertel, G., Stone, D., Johnson, R., and Passmore, J., Eds. *The Wiley-Blackwell Handbook of the Psychology of the Internet at Work*. John Wiley & Sons, Inc., Hoboken, NJ, 2017.
- Kidwell, R.E. and Sprague, R. Electronic surveillance in the global workplace: Laws, ethics, research, and practice. *New Technology, Work, and Employment* 24, 2 (July 2009), 194–208.
- Kühnle, H., Ed. *Distributed Manufacturing: Paradigm, Concepts, Solutions and Examples*. Springer, London, U.K., 2010.
- Lu, J.L. Perceived job stress of women workers in diverse manufacturing industries. *Human Factors and Ergonomics in Manufacturing* 15, 3 (Summer 2005), 275–291.
- Lyytinen, K. and Yoo, Y. Issues and challenges in ubiquitous computing. *Commun. ACM* 45, 2 (Feb. 2002), 63–65.
- McNall, L.A. and Stanton, J.M. Private eyes are watching you: Reactions to location-sensing technologies. *Journal of Business & Psychology* 26, 3 (Sept. 2011), 299–309.
- Meeker, M. *Internet Trends 2017*. Kleiner, Perkins, Caulfield, and Byers, May 31, 2017; <http://dq756f9pzlyr3.cloudfront.net/file/Internet+Trends+2017+Report.pdf>
- Miles, J. and Hollenbeck, J.R. Teams and technology. In *The Psychology of Workplace Technology*, M.D. Covert and L.F. Thompson, Eds. Routledge, New York, 2014, 99–117.
- Monga, V. The new bookkeeper is a robot. *The Wall Street Journal* (May 5, 2015), B1–B7; <https://www.wsj.com/articles/the-new-bookkeeper-is-a-robot-1430776272>
- National Research Council. *Collaboratories: Improving Research Capabilities in Chemical and Biomedical Sciences*. National Academies Press, Washington, D.C., 1999.
- Nixon, A.E. and Spector, P.E. The impact of technology on employee stress, health, and well-being. In *The Psychology of Workplace Technology*, M.D. Covert and L.F. Thompson, Eds. Routledge, New York, 2014, 238–260.
- Orlikowski, W.J. and Scott, S.V. Sociomateriality: Challenging the separation of technology, work, and organization. *Annals of the Academy of Management* 2, 1 (Jan. 2008), 433–474.
- Osman, M. Controlling uncertainty: A review of human behavior in complex, dynamic environments. *Psychological Bulletin* 136, 1 (2010), 65–86.
- Pfanner, E. Japanese robot maker Fanuc reveals some of its secrets. *The Wall Street Journal* (Mar. 27, 2015), B1–B2; <https://www.wsj.com/articles/japanese-robot-maker-fanuc-reveals-some-of-its-secrets-1427384420>
- Potosky, D. and Lomax, M.W. Leadership and technology: A love-hate relationship. In *The Psychology of Workplace Technology*, M.D. Covert and L.F. Thompson, Eds. Routledge, New York, 2014, 118–146.
- Redden, E.S., Elliott, L.R., and Barnes, M.J. Robots: The new teammates. In *The Psychology of Workplace Technology*, M.D. Covert and L.F. Thompson, Eds. Routledge, New York, 2014, 185–208.
- Rosen, L.D., Cheever, N., and Carrier, M., Eds. *The Wiley Handbook of Psychology, Technology and Society*. John Wiley & Sons, Inc., Hoboken, NJ, 2015.
- Ross, P., Sepper, R., and Pohjonen, H. Cross-border teleradiology: Experience from two international teleradiology projects. *European Journal of Radiology* 73, 1 (Jan. 2010), 20–25.
- Rotman, D. How technology is destroying jobs. *MIT Technology Review* (June 12, 2013); <http://www.technologyreview.com/featuredstory/515926/how-technology-is-destroying-jobs/>
- Smite, D., Moe, N.B., and Agerfalk, P.J., Eds. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Springer, Berlin, Germany, 2010.
- Smith, A. and Anderson, J. *AI, Robotics, and the Future of Jobs*. Pew Research Center, Washington, D.C., 2014; <http://www.pewInternet.org/2014/08/06/future-of-jobs>
- Van Hoose, D.D. *E-Commerce Economics*. Routledge, Milton Park, U.K., 2011.
- Vanderkam, L. Work/life integration is the new normal. *Fortune* (Mar. 15, 2015), 139.
- Woodbridge, A. The Icarus syndrome meets the wearable revolution. *Korn/Ferry Briefings on Talent and Leadership* (Feb. 2015), 27–33; <https://www.kornferry.com/institute/icarus-syndrome-meets-wearable-revolution>
- Zuboff, S. *In the Age of the Smart Machine: The Future of Work and Power*. Basic Books, New York, 1988.

Ramiro Montealegre (Ramiro.Montealegre@colorado.edu) is an associate professor of management and entrepreneurship in the Leeds School of Business at the University of Colorado, Boulder.

Wayne F. Cascio (wayne.cascio@ucdenver.edu) is a distinguished professor at the University of Colorado. He holds the Robert H. Reynolds Chair in Global Leadership in the Business School at the University of Colorado, Denver.

Development of energy-efficient software is hindered by a lack of knowledge and a lack of tools.

BY GUSTAVO PINTO AND FERNANDO CASTOR

Energy Efficiency: A New Concern for Application Software Developers

THE PREVALENCE AND ubiquity of mobile computing platforms, such as smartphones, tablets, smart watches, and smart glasses, have changed the way people use and interact with software. In particular, these platforms share a common yet challenging requirement: they are battery-driven. As users interact with them, they tend to be less available, since even simple, well-optimized operations (for example, texting a friend) consume energy. At the same time, wasteful, poorly optimized software can deplete a device's battery much faster than necessary. Heavy resource usage has been shown to be a reason leading to poor app reviews in online app stores.²²

This concern, however, pertains not only to mobile platforms. Indeed, big players in the software industry are also reaching the same conclusion, as stated in one of the very few energy-efficient software development guides: "Even small inefficiencies in apps add up across the system, significantly affecting battery life, performance, responsiveness, and temperature."^a Corporations that maintain datacenters struggle with soaring energy costs. These costs can be attributed in part to overprovisioning with servers constantly operating under their maximum capacity (for example, U.S. datacenters are wasting huge amount of energy¹⁵), and to the developers of the apps running on these datacenters generally not taking energy into consideration.³⁶

Unfortunately, during the last decades, little attention has been placed on creating techniques, tools, and processes to empower software developers to better understand and use energy resources. As a consequence, software developers still lack textbooks, guidelines, courses, and tools to reference when dealing with energy consumption issues.^{36,45} Moreover, most of the research that connects computing and energy efficiency has concentrated on the lower levels of the hardware and software stack. However, recent stud-

a https://developer.apple.com/library/content/documentation/Performance/Conceptual/powerefficiencyguidelines/osx/index.html#apple_ref/doc/fuid/TP40013929

» key insights

- **Developers currently do not fully understand how to write, maintain, and evolve energy-efficient software applications.**
- **Two main problems are identified: Developers lack knowledge on how to measure, profile, and optimize energy efficiency, and they lack tools to help them in these tasks, in particular, tools that work with abstractions they are familiar with.**
- **Software energy consumption research is evolving to mitigate these problems and this article highlights promising research avenues.**

ies show these lower-level solutions do not capture the whole picture^{2,9,25} when it comes to energy consumption. Although software systems do not consume energy themselves, they affect hardware utilization, leading to indirect energy consumption.

How is software related to energy consumption? Energy consumption E is an accumulation of power dissipation P over time t , that is, $E = P \times t$. Power P is measured in watts, whereas energy E is measured in joules. As an example, if one operation takes 10 seconds to complete and dissipates five watts, it consumes 50 joules of energy. In particular, when taking about software energy consumption, one should pay attention to:

- ▶ a given software system under execution,
- ▶ on a given hardware platform,
- ▶ on a given context, and
- ▶ during a given time.

To understand the importance of a *hardware platform*, consider an application that uses the network. Any commodity smartphone today supports, at least, WiFi, 3G, and 4G. A recent study observed that 3G can consume about 1.7x more energy than WiFi, whereas 4G can consume about 1.3x more energy than 3G, while performing the same task, on the same hardware platform.²³

Context also plays a key role, since the way software is built and used has a critical influence on energy consumption. For instance, software can stress energy consumption on CPUs, when performing CPU-intensive computations,⁴⁶ on DRAMs, when performing random accesses to data structures,³⁴ on networks, when running several HTTP requests,^{9,28} and on displays, when using lighter backgrounds^{29,32} or playing videos.

Finally, *time* plays a key role in this equation. A common misconception among developers is that reducing execution time also reduces energy consumption,^{36,45} the t of the equation. However, chances are this reduction in execution time might increase the number of CPU cycles (for example, using multi-core CPUs) and, therefore, the



number of context switches. This, in turn, might increase the P of the equation, impacting the resulting energy consumption.

Software engineering meets energy consumption. While the strategy of leaving the energy consumption optimization problem to the lower-level layers has been successful, recent studies show that even better energy savings can be achieved by empowering and encouraging software developers to participate in the process.^{9,23,34,42} However, the application level, which is the focus of most mainstream software being developed these days, has been the target of few studies.

This lack of evaluation was observed in a recent paper,⁴⁸ where the authors surveyed the papers published during a period of 10 years in top software engineering venues, and found only 20 research papers that have “power” or “energy” on their titles or abstracts. More interestingly, however, the authors observed that none of them were published before 2012. In 2012, three papers were published, whereas six papers were published in 2013 and 11 pa-

pers in 2014. That shows the emerging character of the field.

The need for studies that focus on the higher levels of the software stack is important from at least two important perspectives:

Software engineer’s perspective. Battery usage is a key factor for adopting and evaluating mobile applications. Users of an energy-inefficient app might review it badly, encouraging other users not to use it. This can negatively impact the app’s revenue.

End user’s perspective. The last mile in energy efficiency comes from the choices of end users. To make better choices, and further minimize energy consumption, end users should be aware of the different energy characteristics of software applications that serve the same purpose.

This article is a review of the most prominent software engineering approaches for writing, maintaining, and evolving energy-efficient software applications. We organize the contributions according to the *Guide to the Software Engineering Body of Knowledge (SWE-BOK)*,¹ a common practice in software

engineering studies (for example, Murphy-Hill et al.³⁹). When conducting such review, we found the literature does not cover effectively certain areas of the SWEBOK. For these cases, we share our visions of possible research avenues that energy-aware researchers can follow to reduce this gap.

We unveil the perceptions of mobile developers when dealing with energy consumption issues, scratching their problems and possible solutions. We acknowledge that most of the energy-related problems, in fact, can be reduced to two main problems: the lack of knowledge and the lack of tools and present recent literature to understand how software engineering researchers are tackling these two problems.



A Formative Study

Energy consumption issues are now knocking on the door of application software developers. To shed light on this matter, similarly to Pang et al.,⁴² we conducted a survey with software developers to understand their perceptions about software energy consumption issues. Compared to previous research, which surveyed a wide range of software developers, our target population is more focused and consists of 62 software developers who have performed at least one commit to a mobile open source application.

Among the respondents, 68.75% have more than eight years of software development experience, 57.81% have more than two years of mobile development experience, and 77.41% have more than two years of open source development experience. The majority of them (57.8%) are source code contributors or project owners (35.9%). More interestingly, 70.31% of the respondents agree that energy consumption could be an issue in their mobile applications. Also,

37 respondents have already faced energy-related problems, as a respondent said: “We have a limited energy envelope for the whole system and we must make sure even our power hungry components don’t cause the system to go beyond this limit.” Also, some respondents are aware that energy inefficiencies can impact on app popularity and, therefore, revenue: “Users will leave bad reviews if you drain the battery.”

When asked if they found the root cause for the energy-related problems, 50% of the respondents did not answer. For those who answered, background activities, GPS, and unnecessary resource usage are among the recurring answers. Interestingly, these problems were also observed in other studies.^{36,45} However, 31.81% of the respondents did not observe any significant improvement in energy consumption after applying their solutions. For those who observed an improvement, only five of them made use of specialized tools. The majority of them have the perception of an improvement, for example: “The battery is lasting longer,” “Less heat from device,” or “I really do not measure before and after. It’s just a perception.” When we asked where they find reliable information about what solutions can be used to save energy, seven of them refer to the official documentation, five of them use Stack-Over flow, and five use other channels (blogs, YouTube, open source repositories). Unfortunately, the solutions described in such sources of documentation often are not supported by empirical evidence.^{38,45} To make the matter worse, two respondents rely on “trial and error,” which is far from accurate.

Moreover, 67% of the respondents said that energy-related features are “important” or “very important” to have in well known IDEs. Only eight of the overall respondents have actually used software energy consumption tools. Respondents said that the most important energy-related features to have in well-known IDEs are profiling tools (16 answers), varying from CPU, network, method, wake locks, thread, and live profile. Indeed, one respondent synthesizes that well-known IDEs, such as Android Studio, lack these features: “Android Studio needs a good energy profiler to check the Android power consumption from all power consum-

ers (radios, CPU, memory, storage, everything).” These results not only corroborate with the findings of Pang et al.,⁴² but also reinforce that application-level energy management is in high demand among application software developers, although better support is urgently needed.

We also asked five leading researchers in the area of software energy consumption to identify the most significant contributions and biggest open challenges in this area. All the researchers agreed that tool support is still lacking when it comes to energy measurement, reengineering, refactoring, and other related activities. Even though there is a recent interest from IDE builders to provide an energy consumption perspective of the software systems under development,^b this finding suggests there is still much to do.

Energy-Related Problems

As observed in our formative study, software developers currently have to rely on Q&A websites, blog posts, or YouTube videos when trying to optimize energy consumption, which are anecdotal, not supported by empirical evidence, or even incorrect.^{24,36} The consequence of the lack of appropriate textbooks, guidelines, and cookbooks for green software development is the *lack of knowledge* on how to write, maintain, and evolve energy-efficient software applications. Furthermore, our respondents also mentioned they believe that energy-related features are very important to have in well-known IDEs. In particular, energy profiling techniques can be very helpful. This lack of energy-related features incurs in the *lack of tools* to find, refactor, and fix energy-inefficient code.

The lack of knowledge and the lack of tools to write energy-efficient software is also discussed in the literature. For instance, Pinto et al.⁴⁵ noticed that a common misconception is to confuse concepts such as “power” and “energy.” Manotas et al.³⁶ observed that developers believe in panaceas, that is, solutions that are presented as universal but, in fact, only work in specific contexts. For instance, while one developer

b <https://developer.apple.com/library/ios/documentation/Performance/Conceptual/EnergyGuide-iOS-MonitorEnergyWithXcode.html>

suggested, “offloading computation to the cloud” as a way to improve energy consumption, another developer mentioned, “decreased radio use increases battery life.” As a result, developers should consider the underlying thresholds to take proper advantage of each solution. These are examples of lack of knowledge. To further complicate matters, optimizing performance does not always help to save energy.^{25,26,31,46} Thus, the extensive performance textbooks and guidelines are not always useful.

The aforementioned lack of knowledge is intrinsically connected to the lack of tools. Moura et al.³⁸ observed that energy-aware developers often employ low-level solutions that sometimes result in hard-to-detect correctness problems. The following commit message provides an example of a correctness problem: “Disable Auto Power Saving when resetting the modem. This can cause several bugs with serial communication.”^c High-level energy saving tools might be useful in mitigating this problem. In addition, Pang et al.⁴² found that 88% of the respondents of their survey do not know what tool they can use to measure the energy consumption of their software. These are examples of lack of tools. Although software energy consumption tools do exist, they have yet-to-be-addressed limitations:

- ▶ They require an in-depth knowledge of low-level implementation details and programmers under time pressure have little chance to learn how to use them;
- ▶ They do not provide direct guidance on energy optimization, that is, bridging the gap between understanding where energy is consumed and understanding how the code can be modified in order to reduce energy consumption.

Here, we discuss how current software engineering research is addressing these two key problems.

Energy-Related Solutions

Since there is no single solution for conserving energy, we organize the contributions in terms of the topics of the SWEBOK,¹ a common practice in software engineering studies (for example, Murphy-Hill et al.³⁹). Although energy

Energy consumption issues are now knocking on the door of application software developers.

consumption can be related to any software engineering topic, we chose to focus only on topics directly related to software coding, since it is one of the main activities of software developers, and it is the target of most of the recent research contributions. Therefore, we do not cover the following topics: software configuration management, software engineering management, software engineering process, and software requirements.

Software tools and methods. We organize our discussion of software engineering tools and methods in terms of enhancement methods, measurement tools, and static analysis tools.

Enhancement methods. These methods refer to energy-saving techniques that developers can use, even though they have no prior knowledge of the application domain. For instance, software developers often leverage modern CPUs to dynamically change their operating frequencies, thus reducing power dissipation.³⁸ However, when applying this technique, software developers should use low-level system interfaces that are error-prone and platform dependent. Notwithstanding, blindly downscaling CPU frequency might increase energy consumption while reducing performance.^{20,34} This is an important example of the lack of tools. To mitigate this problem, novel approaches are based on dynamic adaptation through an energy profiler module, energy policies, and energy adaptation APIs.^{49,50} The energy profiler module can recognize the system states and estimate the energy potentially demanded by an application.

Another example is method reallocation,¹⁰ which refers to the analysis of a software system considering all the levels of the stack (for example, kernel, library, and source code level), and reorganizing the classes and methods through the levels of the stack, in a way in which they can be placed in the level where the energy consumption is minimal. As a limitation, this technique can be utilized only if the operating system and the software development environment allow application software developers to go through the different levels (for example, from source code level to kernel level). In a similar strategy, cloud offloading²³ is a technique in which heavy computations are sent


^c <https://github.com/alobo/SerialGSM/commit/c616b950>

to a remote computer; after the remote execution the result is sent back to the local machine. This approach aims to re-organize the implementation of the system at the source code level, thus saving energy by minimizing processing. Interestingly, when we asked the respondents if they found any solution to overcome the energy-related problems, one of the respondents said: “Offload intensive work to workers in the cloud.” However, this technique is only effective if the savings can compensate the extra energy toll required to send a computation through a network. Therefore, trade-offs exist and, as discussed previously, different components have different energy usage characteristics.


Measurement tools. Some measurement tools include methods that use data collected from different system interfaces to assess the energy consumption at the application level. One example is the Running Average Power Limit (RAPL). This module enables architectures monitor energy consumption and store it in Machine-Specific Registers (MSRs).^d Several energy-consumption studies are based on this module (for example, Lima et al.,³⁰ Liu et al.,³⁴ Pinto et al.⁴⁷). With such techniques, it is possible to profile a system and analyze, for instance, what are the system calls that have a major contribution to power dissipation.^{10,34} System calls, in particular, are being actively used for predicting and estimating energy consumption of a software system.^{2,3,8}

Other tools leverage energy models. This strategy utilizes a model developed by physically measuring the energy consumption of a device.^{17,23,26} Energy models have a higher level of confidence only when approximating the energy consumption on the hardware based on which the model was created. Other hardware architectures can only consider the model as a rough estimation.

Although there are already some software tools for energy measurement (for example, Hindle et al.¹⁷ and Li et al.²⁶), such tools have well-known drawbacks. First, energy measurement tools may pay an additional overhead on energy consumption, mostly due to the sampling mechanism. Data acquisition (that is, sampling) is the result of the process of acquiring information from



Interestingly, when we asked the respondents if they found any solution to overcome the energy-related problems, one said: “Offload intensive work to workers in the cloud.”



the surrounding environment, processing the data, and sending it to another collection point to be consumed. Therefore, sampling techniques might impact energy consumption. This poses a challenge, since a recent study provides evidence that a high sampling rate is necessary to obtain reliable information.⁵¹ Even though this problem can be circumvented by employing software-based measurement approaches,³⁴ these approaches are often regarded as less rigorous than hardware-based ones.

Second, hardware- and software-based approaches often do not provide the granularity level that application software developers are interested in.^{36,45} For instance, there is no tool support to measure energy consumption per thread per system module. It is difficult to link the energy measurements across the running threads with fine-grained events that happen during program execution, such as method calls. To make matters worse, the tail energy—that is, the high power state that remains long after the usage of a hardware component, such as the GPS²⁶—should be taken into consideration, even in the presence of context switches. As a result, there is a mismatch between the noise introduced by coarse-grained measurements and the tiny energy impact of methods calls. Still, in our survey, 11 respondents mentioned that measurement tools are among the most important energy-related features to have available in well-known IDEs.

Static analysis tools. One of the main challenges of software energy consumption research is to bring analysis to the static level. Currently, software energy consumption instrumentation can only be conducted at runtime. This approach has several limitations; such as sophisticated (and expensive) hardware equipment⁴⁶ or applicability only to specific hardware configurations.³⁴ This fact has the potential of limiting the usability of software energy consumption tools.

Although there are few studies in this direction (for example, a static analysis technique for estimating the energy consumption of embedded programs³³), these tools often combine static analysis with dynamic analysis techniques (for example, Li et al.^{26,28}), which makes them hardware-depen-

d <https://01.org/msr-tools/overview>

dent, and do not exhibit maturity, nor the breadth of scope necessary for use in real software development. One of the main challenges for deriving static analysis tools for energy consumption is the need for a body of knowledge on how language constructs and design decisions impact energy consumption. Due to the emerging character of the field,⁴⁸ we believe that new empirical energy consumption studies will be conducted in the following years, which in turn will help researchers to create such static analysis tools.

Software maintenance. We organize our discussion of software maintenance in terms of refactoring, reengineering, and visualization.

Refactoring. Refactoring tools can take advantage of cutting-edge research and incorporate such knowledge into refactoring engines. However, as a researcher respondent said, “There is a lot of work showing how different programming styles, techniques, structures influence the consumption, but there is still no real cataloging ... based on these concrete software practices.” Although researchers have been speculating on this subject during the last years,¹⁴ to the best of our knowledge, there is only a handful of studies that deals with the problem of introducing novel refactoring tools for improving the energy efficiency of a software system.^{5,12} In one of these studies, the authors present a set of energy efficiency guidelines that are specifically tailored for Android apps, such as location updates and resource leaks. When applied, the authors observed improvements of up to 29% of the overall energy consumption.

This lack of contributions is not related to a lack of opportunities. As mentioned, there are several opportunities for application software developers to save energy by refactoring existing systems.^{19,48} For example, Pinto et al.⁴⁷ observed that just updating from Hashtable to Concurrent HashMap in a Java program can yield a 3.5x energy savings. In particular, this transformation yields a 1.4x and a 9.2x energy savings in CPU and DRAM, respectively. As another example, Pathak et al.⁴³ observed that I/O operations consume more energy partly because of the tail energy phenomenon. According to the authors, bundling I/O operations together can mitigate this tail energy leak. These re-

sults have a clear implication: Tools to aid developers in quickly refactoring programs can be useful if energy is important.

Reengineering. Compared to refactoring tools, which are more localized, reengineering efforts can be broader in scope and have a systemwide impact on the structure of an application. As mentioned, method reallocation¹⁰ and method offloading²³ are two common strategies to implement reengineering energy-aware methods. This is corroborated by the work of Othman et al., which found that up to 20% energy savings can be achieved by uploading tasks from mobile devices to fixed servers.⁴¹ Using a different strategy, Manotas et al.³⁷ proposed SEEDS, a general decision making framework for optimizing software energy consumption. The SEEDS framework can identify energy-inefficient uses of Java collections, and automate the process of selecting more efficient ones. Similarly, Fernandes et al.¹³ developed a tool that leverages static and dynamic analysis to recommend the most energy-efficient data structures. Search-based software engineering approaches were used to reengineer a software system in order to minimize energy usage,⁶ yielding an energy reduction of up to 25%. These approaches mitigate the problem of lack of tools.

Visualization techniques are useful to support the understanding of software systems in order to discover and analyze their anomalies. Li et al.²⁶ proposed a technique that overlays energy consumption information with application’s source code. This technique colors different amount of energy consumed in a given line of code—blue lines describe low energy consumption whereas red lines indicate high-energy consumption. This visualization technique is fine-grained and works at the source code level. On the other hand, the study of Couto et al.¹¹ focuses on a coarser granularity: It identifies the energy consumption per method, and aggregates this energy in terms of classes, packages, and the whole software system. The result is presented in a sunburst diagram that allows developers to easily and quickly identify the most energy-inefficient parts of the code. These studies combine art and technology as a way to represent energy consump-

tion. With a better understanding of the whole program energy behavior, such visualization techniques can be useful to mitigate both lack of knowledge and lack of tools.

Software design and construction. Researchers have been studying different strategies for designing and constructing energy-efficient software.^{16,25,29,31,43} These studies focus on understanding how a particular programming practice or design implementation might impact on energy consumption. To gain further confidence in the results, these studies often analyze dozens (for example, Kambadur²⁰), or even hundreds (for example, Li et al.²⁵), of software applications, and they mitigate the lack of knowledge by providing high-level guidelines for designing energy-efficient software. We organize our discussions of software design and construction in terms of mobile, network, data structures, and parallel programming techniques.

Mobile development. Linares-Vasquez et al.³¹ investigated API calls that might cause high-energy consumption. For example, they observed that the method `Activity.findViewById`, which is commonly used, is one of the most energy-consuming among the Android APIs. Similarly, Malik et al.³⁵ found that the `BroadcastReceiver` and the `Location` APIs are the most often discussed among Android energy questions on StackOver flow. Furthermore, since the display is one of the smartphone’s most energy-intensive components,⁷ Li et al.²⁹ discussed how to improve energy efficiency by favoring darker colors instead of lighter ones for smartphones with OLED displays. Using a search-based multi-objective approach, Linares-Vasquez et al.³² automatically optimized energy consumption and contrast, while using consistent colors with respect to the original color palette. Oliveira Jr. et al.¹⁹ analyzed the energy consumption of Android app development approaches, Java, JavaScript, and Java + C++, in both benchmarks and real apps. In both scenarios it was observed that different approaches have different impacts on energy. In particular, combining different approaches can yield more than an order of magnitude energy savings in compute-intensive apps.

Network usage. Li et al.²⁵ analyzed

more than 400 real-world Android apps, and found that an HTTP request is the most energy-consuming operation of the network. In a follow-up study, the same authors observed that bulking HTTP requests is a good practice for energy saving.²⁸ Also regarding HTTP usage, Chowdhury et al.⁹ observed that HTTP/2 is more energy efficient than its predecessor, HTTP/1.1, for networks with higher Round Trip time (RTTs). Since most mobile apps use network,²⁵ we expect more contributions on this direction. Besides of bulking requests, researchers can evaluate the benefits of, for instance, reducing transactions, compressing data, and appropriately handling errors to conserve energy.

Data structures. The energy behavior of different data structures, one of the building blocks of computer programming, have been extensively studied in the last few years.^{16,30,37,47} Hasan et al.¹⁶ investigated data structures grouped with three interfaces (List, Set, and Map). Among the findings, they found that the position where an element is inserted in a list can greatly impact energy consumption. Pinto et al.⁴⁷ studied the same group of interfaces, but focused on thread-safe data structures. They also observed that using a newer version of a thread-safe data structure can yield a 2.19x energy savings when compared to the old associative implementation. Lima et al.³⁰ studied the energy consumption of data structures in concurrent functional programs. Although they found that there is no clear universal winner, in certain circumstances, choosing one data sharing primitive (MVar) over another (TMVar) can yield 60% energy savings.

Parallel programming. Parallel programming techniques have also been the subject of several studies. Pinto et al.⁴⁶ observed that a high-level, work-stealing parallel framework is more energy-friendly when performing fine-grained CPU intensive computations than a thread-based implementation. Still, Ribic and Liu proposed a set of runtime systems for improving the energy efficiency of fine-grained CPU-intensive computations.^{49,50} To better leverage the energy savings reported by these studies, we believe they can be integrated with well-known runtime systems, such as the Java Virtual Machine (JVM). If



so, the whole chain of programming languages, software systems, and end-users that rely on the JVM can benefit from these findings.

Although these studies provide a comprehensive set of findings with practical and timely implications and can be useful to mitigate the problem of lack of knowledge, they are far from covering the whole spectrum of programming language constructs and libraries.

Software quality and testing. Here we organize our discussions in terms of software testing and software debugging techniques.

Software testing. Although there are several studies aimed at characterizing energy bugs (for example, Pathak et al.⁴⁴), there are relatively few studies that propose new energy-aware testing techniques.^{18,21,27} Ding et al.²⁷ presented an energy-efficient testing suite minimization technique that can be used to perform post-deployment testing on embedded systems. Results suggest the approach can promote a reduction of over 95% of the energy consumed by the original test suite. Similarly, Jabbarvand et al.¹⁸ present another test suite minimization approach, but focusing on Android apps. The authors reported a reduction of, on average, 84%, while maintaining the effectiveness for revealing bugs. Kan²¹ proposes a similar approach: To use DVFS to scale frequency down when running the test suites. Although some researchers argued that DVFS tech-

niques can lead to increased energy consumption and performance loss,³⁴ the authors showed that important energy savings can be achieved. Banerjee et al.⁴ proposed a technique that generates test inputs that are likely to capture energy bugs. This technique focuses on creating tests that use I/O components, which are one of the primary sources of energy consumption in a smartphone.^{7,43}


Followed by these promising initial results, we believe that new testing techniques will be evaluated in terms of energy consumption. At best, energy testing will become a research area. Several possible areas of interest can be envisioned. One of them is what we call “green assertions,” that is, the possibility to define an energy budget where the test case asserts whether the computation satisfies that budget. The test fails if the energy consumed is greater than the suggested budget. For instance, the code snippet `double maxEnergy = 200; assertTrue(render(), expected, maxEnergy);` defines that the `render()` method should consume, at most, 200 Joules. This technique can be further improved to cover additional hardware characteristics, for instance, asserting whether the computation consumes 100 Joules due to network communication or 50 Joules due to the CPU.

Software debugging. Practitioners commonly use debugging tools to catch bugs in program formulation. However, debugging an energy-inefficient piece of code is more challenging than traditional debugging because such inefficiencies depend on the contextual information about where a program is running, such as the state of the hardware devices. In this regard, Banerjee and colleagues⁵ propose a framework for debugging energy consumption-related field failures in mobile apps. The authors found that tool support could localize energy bugs in a short amount of time, even for non-trivial Android apps. The authors observed energy savings of up to 29% after patching the energy bug. Pathak et al.⁴³ propose *eprof*, a fine-grained profiling energy consumption technique for applications running on smartphones. Similar to the work of Banerjee and colleagues,⁴ Pathak et al. focus on understanding and monitoring system calls

that are related to I/O operations. As a results, they found that most of the energy consumed in free apps is related to third-party advertisement modules (which can be responsible for up to 75% of the overall energy consumed by an app). Using a collaborative black-box approach, Oliner et al⁴⁰ propose a method for diagnosing anomalies, estimating their severity, and identifying the device features that lead to the anomaly. Using feedback received by the proposed tool, end users improved their battery life by 21%.

We believe that debugging tools will have the capability of inspecting the energy consumption of fine-grained program constructs during runtime, as well as their common ability to identify which value was attributed to a given variable. Debugging tools can go further and highlight the CPU intensive lines of code, or the memory-intensive methods, in a way that developers can refactor them in an energy-savvy manner. Novel energy-related testing and debugging tools can mitigate the lack of tools.

Conclusion

Energy consumption is a ubiquitous problem and the years to come will require developers to be even more aware of it. However, developers currently do not fully understand how to write, maintain, and evolve energy-efficient software systems. In this study we suggest this is primarily due to two problems: the lack of knowledge and the lack of tools. With these problems in mind, this article reviewed most of the recent energy-related contributions in the software engineering community. We discuss how software energy consumption research is evolving to mitigate these two problems and, when appropriate, we highlight key research gaps that need better attention. 

References

1. Abran, A., Bourque, P., Dupuis, R., and Moore, J.W., editors. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 2001.
2. Aggarwal, K., Hindle, A., and Stroulia, E. GreenAdvisor: A tool for analyzing the impact of software evolution on energy consumption. In *Proceedings of ICSME* (Sept. 2015), 311–320.
3. Aggarwal, K., Zhang, C., Campbell, J.C., Hindle, A., and Stroulia, E. The power of system call traces: Predicting the software energy consumption impact of changes. In *Proceedings of CASCON*, 2014, 219–233.
4. Banerjee A., Chong, L.K., Chattopadhyay, S., and Roychoudhury, A. Detecting energy bugs and hotspots in mobile apps. In *Proceedings of ESEC/FSE*, 2014, 588–598.
5. Banerjee, A. and Roychoudhury, A. Automated refactoring of android apps to enhance energy-efficiency. In *Proceedings of MOBILESofT*, 2016, 139–150.
6. Bruce, B.R., Petke, J., and Harman, M. Reducing energy consumption using genetic improvement. In *Proceedings of GECCO*, 2015, 1327–1334.
7. Carroll, A. and Heiser, G. An analysis of power consumption in a smartphone. In *Proceedings of USENIX*, 2010.
8. Chowdhury, S.A. and Hindle, A. Greenoracle: Estimating software energy consumption with energy measurement corpora. In *Proceedings of MSR*, 2016, 49–60.
9. Chowdhury, S.A., Sapra, V. and Hindle, A. Client-side energy efficiency of HTTP/2 for web and mobile app developers. In *Proceedings of SANER*, 2016, 529–540.
10. Corral, L., Georgiev, A.B., Silitti, A., and Succi, G. Method reallocation to reduce energy consumption: an implementation in android OS. In *Proceedings of SAC*, 2014, 1213–1218.
11. Couto, M., Carcao, T., Cunha, J., Fernandes, J.P., and Saraiva, J. Detecting anomalous energy consumption in android applications. In *Proceedings of SBLP*, 2014, 77–91.
12. Cruz, L., Abreu, R., and Rouvignac, J.-N. Leafactor: Improving energy efficiency of android apps via automatic refactoring. In *Proceedings of MobileSoft*, (Buenos Aires, Argentina, 2017).
13. Fernandes, B., Pinto, G., and Castor, F. Assisting non-specialist developers to build energy-efficient software. In *Proceedings of the Companion to the 39th International Conference on Software Engineering*, (Buenos Aires, Argentina, 2017).
14. Fraser, S., Murphy-Hill, E., Wild, W., Yoder, J., and Zhu, B.Q. Going green with refactoring: Sustaining the 'worldwide virtual machine'. In *Proceedings of OOPSLA*, 2011, 171–174.
15. Gelenbe, E. and Caseau, Y. The impact of information technology on energy consumption and carbon emissions. *Ubiquity*, (June 2015) 1–15.
16. Hasan, S., King, Z., Hafiz, M., Sayagh, M., Adams, B., and Hindle, A. Energy profiles of Java collections classes. In *Proceedings of ICSE*, 2016, 225–236.
17. Hindle, A., Wilson, A., Rasmussen, K., Barlow, E.J., Campbell, J.C., and Romansky, S. Greenminer: A hardware based mining software repositories software energy consumption framework. In *Proceedings of MSR*, 2014, 12–21.
18. Jabbarvand, R., Sadeghi, A., Bagheri, H., and Malek, S. Energy-aware test-suite minimization for android apps. In *Proceedings of ISSTA*, 2016, 425–436.
19. Oliveira Jr., W.O.R., and Castor, F. A study on the energy consumption of android app development approaches. In *Proceedings of MSR* (Buenos Aires, Argentina, 2017).
20. Kambadur, M. and Kim, M.A. An experimental survey of energy management across the stack. In *Proceedings of OOPSLA*, 2014, 329–344.
21. Kan, E.Y.Y. Energy efficiency in testing and regression testing—A comparison of DVFS techniques. In *Proceedings of QSI'13*, 2013, 280–283.
22. Khalid, H., Shihab, E., Nagappan, M., and Hassan, A.E. What do mobile app users complain about? *IEEE Software* 32, 3 (2015), 70–77.
23. Kwon, Y. and Tilevich, E. Reducing the energy consumption of mobile applications behind the scenes. In *Proceedings of ICSM*, 2013, 170–179.
24. Li, D. and Halfond, W.G.J. An investigation into energy-saving programming practices for android smartphone app development. In *Proceedings of GREENS* 2014, 46–53.
25. Li, D., Hao, S., Gui, J., and Halfond, W.G.J. An empirical study of the energy consumption of Android applications. In *Proceedings of ICSME*, 2014, 121–130.
26. Li, D., Hao, S., Halfond, W.G.J., and Govindan, R. Calculating source line level energy information for android applications. In *Proceedings of ISSTA*, 2013, 78–89.
27. Li, D., Jin, Y., Sahin, C., Clause, J., and Halfond, W.G.J. Integrated energy-directed test suite optimization. In *Proceedings of ISSTA*, 2014, 339–350.
28. Li, D., Lyu, Y., Gui, J., and Halfond, W.G.J. Automated energy optimization of http requests for mobile applications. In *Proceedings of ICSE*, 2016, 249–260.
29. Li, D., Tran, A.H., and Halfond, W.G.J. Making web applications more energy efficient for OLED smartphones. In *Proceedings of ICSE*, 2014, 527–538.
30. Lima, L.G., Soares-Neto, F., Lieuthier, P., Castor, F., Melfe, G., and Fernandes, J.P. Haskell in green land: Analyzing the energy behavior of a purely functional language. In *Proceedings of SANER*, 2016, 517–528.
31. Linares-Vasquez, M., Bavota, G., Bernal-Cardenas, C., Oliveto, R., Di Penta, M., and Poshyvanyk, D. Mining energy-greedy api usage patterns in android apps: An empirical study. In *Proceedings of MSR*, 2014, 2–11.
32. Linares-Vasquez, M., Bavota, G., Bernal-Cardenas, C., Oliveto, R., Di Penta, M., and Poshyvanyk, D. Optimizing energy consumption of GUIs in android apps: A multi-objective approach. In *Proceedings of ESEC/FSE*, 2015, 143–154.
33. Liqat, U., et al. Energy consumption analysis of programs based on XNOS isa-level models. In *Proceedings of the 23rd International Symposium on Logic-Based Program Synthesis and Transformation*, 2013, 72–90.
34. Liu, K., Pinto, G., and Liu, Y.D. Data-oriented characterization of application-level energy optimization. In *Proceedings of FASE*, 2015.
35. Malik, H., Zhao, P., and Godfrey, M.W. Going green: An exploratory analysis of energy-related questions. In *Proceedings of MSR*, 2015, 418–421.
36. Manotas, I., et al. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of ICSE*, 2016, 237–248.
37. Manotas, I., Pollock, L., and Clause, J. Seeds: A software engineer's energy-optimization decision support framework. In *Proceedings of ICSE*, 2014, 503–514.
38. Moura, I., Pinto, G., Ebert, F., and Castor, F. Mining energy-aware commits. In *Proceedings of MSR*, (2015), 56–67.
39. Murphy-Hill, E., Zimmermann, T., and Nagappan, N. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *Proceedings of ICSE*, 2014, 1–11.
40. Oliner, A.J., Iyer, A.P., Stoica, I., Lagerspetz, E., and Tarkoma, S. Carat: Collaborative energy diagnosis for mobile devices. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, 2013, 10:1–10:14.
41. Othman, M. and Hailes, S. Power conservation strategy for mobile computers using load sharing. *SIGMOBILE Mob. Comput. Commun. Rev.* 2, 1 (Jan. 1998), 44–51.
42. Pang, C., Hindle, A., Adams, B., and Hassan, A.E. What do programmers know about software energy consumption? *IEEE Software* 33, 3 (2016), 83–89.
43. Pathak, A., Hu, Y.C., and Zhang, M. Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof. In *Proceedings of EuroSys*, 2012, 29–42.
44. Pathak, A., Jindal, A., Hu, Y.C., and Midkiff, S.P. What is keeping my phone awake? Characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proceedings of MobiSys*, 2012, 267–280.
45. Pinto, G., Castor, F., and Liu, Y.D. Mining questions about software energy consumption. In *Proceedings of MSR*, 2014, 22–31.
46. Pinto, G., Castor, F., and Liu, Y.D. Understanding energy behaviors of thread management constructs. In *Proceedings of OOPSLA*, 2014, 345–360.
47. Pinto, G., Liu, K., Castor, F., and Liu, Y.D. A comprehensive study on the energy efficiency of Java thread-safe collections. In *Proceedings of ICSME*, 2016.
48. Pinto G., Soares-Neto, F., and Castor, F. Refactoring for energy efficiency: A reflection on the state of the art. In *Proceedings of GREENS*, 2015.
49. Ribic, H. and Liu, Y.D. Energy-efficient work-stealing language runtimes. In *Proceedings of ASPLOS*, 2014, 513–528.
50. Ribic, H. and Liu, Y.D. Aequitas: Coordinated energy management across parallel applications. In *Proceedings of the 2016 International Conference on Supercomputing*, 2016, 4:1–4:12.
51. Saborido, R., Arnaoudova, V., Beltrame, G., Khomh, F., and Antoniol, G. On the impact of sampling frequency on software energy measurements. *PeerJ PrePrints*, 2015, 3:e1219.

Gustavo Pinto (gpinto@ufpa.br) is an assistant professor at the Federal University of Pará, Brazil.

Fernando Castor (castor@cin.ufpe.br) is an associate professor at the Federal University of Pernambuco, Brazil.

©2017 ACM 0001-0782/17/12 \$15.00.



Watch the authors discuss their work in this exclusive *Communications* video. <https://cacm.acm.org/videos/energy-efficiency-a-new-concern-for-application-software-developers>

Introducing *ACM Transactions on Human-Robot Interaction*

Now accepting submissions to ACM THRI

In January 2018, the *Journal of Human-Robot Interaction* (JHRI) will become an ACM publication and be rebranded as the *ACM Transactions on Human-Robot Interaction* (THRI).

Founded in 2012, the *Journal of HRI* has been serving as the premier peer-reviewed interdisciplinary journal in the field.

Since that time, the human-robot interaction field has experienced substantial growth. Research findings at the intersection of robotics, human-computer interaction, artificial intelligence, haptics, and natural language processing have been responsible for important discoveries and breakthrough technologies across many industries.

THRI now joins the ACM portfolio of highly respected journals. It will continue to be open access, fostering the widest possible readership of HRI research and information. All issues will be available on the ACM Digital Library.

Editors-in-Chief Odest Chadwicke Jenkins of the University of Michigan and Selma Šabanović of Indiana University plan to expand the scope of the publication, adding a new section on mechanical HRI to the existing sections on computational, social/behavioral, and design-related scholarship in HRI.

The inaugural issue of the rebranded *ACM Transactions on Human-Robot Interaction* is planned for March 2018.

To submit, go to <https://mc.manuscriptcentral.com/thri>



research highlights

P. 78

**Technical
Perspective
Pricing Information
(and Its Implications)**

By Aaron Roth

P. 79

A Theory of Pricing Private Data

By Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu

P. 87

**Technical
Perspective
A Simple, Elegant
Approach to
Non-Numeric
Parallelization**

By James Larus

P. 88

Automatically Accelerating Non-Numerical Programs by Architecture-Compiler Co-Design

By Simone Campanoni, Kevin Brownell, Svilen Kanev,
Timothy M. Jones, Gu-Yeon Wei, and David Brooks

Technical Perspective

Pricing Information (and Its Implications)

By Aaron Roth

SELLING PERSONAL INFORMATION is very different from selling physical goods, and raises novel challenges. On the sell-side of the market, individuals own their own personal data and experience costs based on the usage of their data insofar as that usage leads to future quantifiable harm. On the buy-side of the market, buyers are interested in “statistical information” about the dataset, that is, aggregate information, rather than information derived from a single individual. Differential privacy¹ provides a means to quantify the harm that can come to individual data owners as the result of the use of their data. This ability to quantify harm allows for data owners to be compensated for the risk they incur. Past work studying markets for private data focused on the simple case in which the buyer is interested in only the answer to a single linear function of the data,^{2,3,4,6} which makes the buy-side of the market particularly simple.

The following paper introduces a fascinating and complicated issue that arises on the buy-side of the market when buyers are interested in multiple linear functions of the same dataset. Information exhibits complementarities: given some information about a dataset, it is possible to *learn* other things about the dataset. This means that when pricing information, there might be opportunities for *arbitrage*: rather than directly buying the answer to the query he is interested in, the buyer might instead more cheaply buy a bundle of queries that lets him deduce the answer he is interested in. The authors give conditions under which a pricing is arbitrage free. This is a compelling condition to ask for: it means that it is a dominant strategy for arriving buyers to faithfully request the answer to the query they are interested in, rather than trying to game the system. By asking for arbitrage-free pricings, the

authors are making the market safe for buyers.


Reasoning about these arbitrage opportunities can be complicated: if the values of the purchased linear functions were revealed exactly, then the answer to any other query in the span of the purchased queries would be derivable. But to guarantee the sellers differential privacy, it is necessary to sell only noisy estimates of the data. This makes reasoning about what is derivable complex. Sensibly, since they are introducing a new problem, the authors opt to study a restricted notion of derivability and arbitrage. They give pricings that rule out arbitrage opportunities when the buyer is only allowed to learn by taking linear combinations of observed queries, is interested only in unbiased estimates of query values, and will attempt arbitrage only at the level of one query at a time. Because of the richness of the authors’ problem, one of the most exciting aspects of this work is the doors it opens for future exploration. Here, I will highlight what I think are the most interesting problems coming out of this paper:

1. Multi-query arbitrage: The paper gives query pricings such that a buyer can never more cheaply derive the answer to a single query by buying a different bundle of queries. However, the buyer can still sometimes more cheaply derive the answer to one *bundle* of queries by buying the answers to another bundle! Which pricings can prevent this?

2. Arbitrage for biased estimators: When buyers are only interested in unbiased estimates, the best linear unbiased estimator is always given by least-squares linear regression. However, buyers can often improve the accuracy of their derivations by trading off a small amount of bias for a large reduction in variance. The space of optimal estimators is much more complex when they do so. In general,

a buyer can access the dataset by asking arbitrary “*statistical queries*,” and can run any learning algorithm in the “SQ model” to derive the answers to other queries.⁵ Can the large literature on SQ learning be used to give arbitrage-free pricings for more general notions of arbitrage?

3. Seller profit: Arbitrage-free pricings give a family of dominant-strategy truthful mechanisms for selling information. Suppose we know something about the distribution of buyer demands—can we find the arbitrage-free pricing that maximizes seller profit? This is particularly intriguing, because it seems the seller can sometimes increase her profit by selling noisier queries, thereby reducing the complementarities among the goods she is selling. The opportunity to increase profit by degrading product quality rarely arises in markets for physical goods.

This paper opens a rich research direction. I recommend that new Ph.D. students (or anyone looking for an attractive problem) read it. 

References

1. Dwork, C., McSherry, F., Nissim, K. and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, 2006, 265–284.
2. Fleischer, L. and Lyu, Y.-H. Approximately optimal auctions for selling privacy when costs are correlated with data. In *Proceedings of the ACM Conference on Electronic Commerce*, 2012, 568–585.
3. Ghosh, A. and Roth, A. Selling privacy at auction. *Games and Economic Behavior* (2015), 334–346.
4. Ghosh, A., Ligett, K., Roth, A. and Schoenebeck, G. Buying private data without verification. In *Proceedings of the ACM Conference on Economics and Computation*, 2014, 931–948.
5. Kearns, M. Efficient noise-tolerant learning from statistical queries. *J. ACM* (1998), 983–1006.
6. Nissim, K., Vadhan, S. and Xiao, D. Redrawing the boundaries on purchasing data from privacy-sensitive individuals. *Innovations in Theoretical Computer Science* (2014), 411–422.

Aaron Roth (aaroth@cis.upenn.edu) is the Class of 1940 Bicentennial Term Associate Professor at the University of Pennsylvania, Philadelphia, PA.

A Theory of Pricing Private Data

By Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu

Abstract

When the analysis of individuals' personal information has value to an institution, but it compromises privacy, should individuals be compensated? We describe the foundations of a market in which those seeking access to data must pay for it and individuals are compensated for the loss of privacy they may suffer.

1. INTRODUCTION

The interests of individuals and institutions with respect to personal data are often at odds. Personal data has great value to institutions: they eagerly collect it and monetize it by using it to model customer behavior, personalize services, target advertisements, or by selling the data directly. Yet the inappropriate disclosure of personal data poses a risk to individuals. They may suffer a range of harms including elevated prices for goods or services, discrimination, or exclusion from employment opportunities.³

A rich literature on privacy-preserving data analysis^{4, 6, 11} has tried to devise technical means for negotiating these competing interests. The goal is to derive accurate aggregate information from data collected from a group of individuals while at the same time protecting each member's personal information. But this approach necessarily imposes restrictions on the use of data. A seminal result from this line of work is that any mechanism providing reasonable privacy must strictly limit the number of query answers that can be accurately released.⁵ Nevertheless, recent research into differential privacy,⁷ a formal model of privacy in which an individual's privacy loss is rigorously measured and bounded, has shown that, for some applications, accurate aggregate analysis need not entail significant disclosure about individuals. Practical adoption of these techniques is slowing increasing: they have been used in a U.S. Census product¹⁶ and for application monitoring by Google⁹ and Apple.¹³

But there remain settings where strictly limiting privacy loss degrades the utility of data and means the intended use of data will be impossible. We therefore pursue an alternative approach which allows a non-negligible degree of privacy loss if that loss is compensated in accordance with users' preferences. Compensating privacy loss is an improvement over the narrower view that mandates negligible privacy loss because it empowers individuals to control their data through financial means and permits more accurate data analysis if end-users are willing to pay for it.

Considering personal information as a resource, one that is valuable but also exchangeable, is not new. Twenty years ago, Laudon proposed that personal information be bought and sold in a national market¹⁸ and there is a mature literature on economic aspects of privacy.¹ And in today's online services, one could argue that individuals are compensated indirectly for contributing their personal data. Many internet

companies acquire personal data by offering a (purportedly) free service, attracting users who provide their data, and then monetizing the personal data by selling it, or by selling information derived from it, to third parties.

Even so, a technical foundation for a market for personal information is lacking, particularly one that is consistent with recent advances in the formal modeling of privacy. We address this by proposing a formal framework for assigning prices to queries in order to compensate data owners for their loss of privacy. Our framework borrows from, and extends, key principles from both differential privacy^{7, 8} and data markets.^{17, 21}

There are three types of actors in our setting: individuals, or data *owners*, contribute their personal data; a *buyer* submits an aggregate query over many owners' data; and a *market maker*, trusted to answer queries on behalf of owners, charges the buyer and compensates the owners.

Our framework makes three important connections:

1.1. Perturbation and price

In response to a buyer's query, the market maker computes the true query answer, adds random noise, and returns a perturbed result. Using differential privacy, perturbation is always necessary. Here query answers can be sold unperturbed, but the price would be high because each data owner contributing to an aggregate query needs to be compensated. By adding perturbation to the query answer, the price can be lowered: the more perturbation, the lower the price. When issuing the query, the buyer specifies the degree of accuracy for which he is willing to pay. Unperturbed query answers are very expensive, but at the other extreme, query answers are almost free if the noise added is the same as required by differential privacy with conservative privacy parameters. The relationship between the accuracy of a query result and its cost depends on the query and the preferences of contributing data owners. Formalizing this relationship is one of the goals of this article.

1.2. Arbitrage and perturbation

Arbitrage allows a buyer to obtain the answer to a query more cheaply than its advertised price by deriving the answer from a less expensive alternative set of queries. Arbitrage is possible because of inconsistency in a set of priced queries. As a simple example, suppose that a given query is sold with two options for perturbation, measured by variance: \$5 for a variance of 10 and \$200 for a variance of 1. A savvy buyer seeking a variance of 1 would never pay \$200. Instead, he would purchase the first query 10 times, receive 10 noisy answers, and compute their average. Since noise is added independently, the variance of the resulting average is 1,

The original version of this paper was published in *ACM Transactions on Database Systems* 39, 4 (Dec. 2014), 1–28.

and the total cost is only \$50. The pricing of queries should avoid arbitrage opportunities. While this has been considered before for data markets,^{2, 17, 21} it has not been studied for perturbed query answers. Formalizing arbitrage for noisy queries is a second goal of this article.

1.3. Privacy-loss and payments

Given a randomized mechanism for answering a query q , a common measure of privacy loss to an individual is defined by differential privacy⁷: it is the maximum ratio between the probability of returning some fixed output with and without that individual's data. Differential privacy imposes a bound of e^ϵ on this quantity, where ϵ is a small constant, presumed acceptable to all individuals. Our framework contrasts with this in several ways. First, the privacy loss is not bounded, but depends on the buyer's request. If the buyer asks for a query with low variance, then the privacy loss to individuals will be high. These data owners must be compensated for their privacy loss by the buyer's payment. In addition, we allow each data owner to value their privacy loss separately, by demanding greater or lesser payments. Formalizing the relationship between privacy loss and payments to data owners is a third goal of this article.

In our framework, the burden of the market maker is not to enforce a strict limit on the privacy loss of individuals. Instead, they must ensure that prices are set such that, whatever disclosure is obtained by the buyer, all contributing individuals are properly compensated. In particular, if a sequence of queries can indeed reveal the private data for most individuals, its price must approach the total cost of the entire database.

2. BACKGROUND

In this section we describe the basic architecture of the private data pricing framework, illustrated as shown in Figure 1.

2.1. The main actors

The main actors in our proposed marketplace are data owners, query buyers, and a market maker negotiating between the two.

The market maker. The market maker is trusted by the buyer and by each of the data owners. He collects data from the owners and sells it in the form of queries. When a buyer decides to purchase a query, the market maker collects payment, computes the answer to the query, adds noise as appropriate, returns the result to the buyer, and finally distributes individual payments to the data owners. The market maker may retain a fraction of the price as profit.

The owner and her data. Each owner contributes a single tuple conforming to a relational schema $R(\mathbb{A})$, with attributes $\mathbb{A} = \{A_1, A_2, \dots, A_k\}$. The crossproduct of the attribute domains, written $dom(\mathbb{A})$, is the set of all possible tuples that could occur. For a fixed k , its size, $n = |dom(\mathbb{A})|$, is polynomial in the number of users; n grows exponentially with k . Having collected tuples from each owner, the market maker forms a relational table I , an instance of $R(\mathbb{A})$.

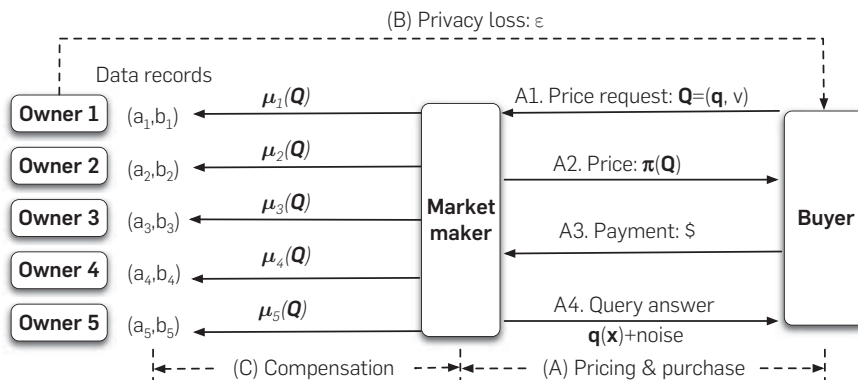
The buyer and his queries. The buyer is a data analyst who wishes to compute some queries over a data. We restrict our attention to linear aggregation queries over instance I , a common class of queries that has received considerable attention from the privacy community.^{8, 15, 19}

To express linear queries, we first represent the instance I by a finite *data vector* \mathbf{x} consisting of non-negative integral counts. For each element $t \in dom(\mathbb{A})$, the vector \mathbf{x} has one entry that reports the number of individuals whose tuple matches t . We assume that $dom(\mathbb{A})$ is ordered, and denote x_i the i 'th entry in the vector \mathbf{x} . In other words, x_i represents the number of individuals whose attribute values match the i 'th entry in $dom(\mathbb{A})$. Although the size n of the vector \mathbf{x} can be large, in practice one avoids fully materializing \mathbf{x} and retains only the relational representation of I .

DEFINITION 1 (LINEAR QUERY). A linear query is a real-valued vector $\mathbf{q} = (q_1, q_2 \dots q_n)$. The answer $\mathbf{q}(\mathbf{x})$ to a linear query on \mathbf{x} is the vector product $\mathbf{q}\mathbf{x} = q_1x_1 + \dots + q_nx_n$.

The class of linear queries includes many common aggregation queries including general predicate counting queries and familiar statistical queries such as histogram counts,

Figure 1: The pricing framework has 3 components: (A) Pricing and purchase: the buyer asks a query $Q = (q, v)$ and must pay its price, $\pi(Q)$; (B) Privacy loss: by answering Q , the market maker leaks some information ϵ about the private data of the data owners to the buyer; (C) Compensation: the market maker must compensate each data owner for their privacy loss with micro-payments; $\mu_i(Q)$ is the total of the micro-payments for all users in bucket i . The pricing framework is *balanced* if the price $\pi(Q)$ is sufficient to cover all micro-payments μ_i and the micro-payments μ_i compensate the owners for their privacy loss ϵ .



data cube queries, and marginals. Linear queries can express differences, weighted counts, and averages over discrete domains. While the query vector \mathbf{q} is large, in practice queries are expressed compactly, directly over the relational representation of a data, in a query language like Structured Query Language (SQL).

EXAMPLE 2. Consider a competition between candidates A and B that is decided by a population of voters. The ratings and descriptive fields of voters are sensitive and should be compensated properly if used in any way. Imagine that users contribute their gender and age along with two numerical ratings, each consisting of values from the domain $\{0, 1, 2, 3, 4, 5\}$. Thus Alice's tuple may be ("Female", 39, 0, 5) if she strongly favors candidate B over candidate A . If the domain for the age attribute is $[1 \dots 120]$, then the database vector will have size $2 \times 120 \times 6 \times 6 = 8640$.

Examples of linear queries include: the total of all ratings for candidate A ; the number of female voters over 40 who gave candidate A a rating of 5; the number of voters whose rating for candidate A exceeds that for candidate B .

We make two comments about our framework. First, although the vector \mathbf{x} removes personally identifiable information (since x_i is the total number of users with a particular combination of attribute values), queries can still leak information about individual users. Differential privacy is designed to limit such leaks. Second, we assume that the buyer is allowed to issue multiple queries, which means that he can combine information derived from multiple queries to infer answers to other queries not explicitly requested. This presents a challenge we must address: to ensure that the buyer pays for any information that he might derive directly or indirectly.

2.2. Balanced pricing frameworks

The market maker enters into two contracts: (1) they promise to answer a buyer's queries according to an agreed price, π , (Section 3), and (2) they promise to compensate the data owners with a micro-payment $\mu_i(\varepsilon)$ whenever they suffer a privacy loss ε in response to a buyer's query (Section 5).

In the contract with the buyer, the market maker allows the buyer to specify, for each linear query \mathbf{q} , an amount of noise ν that he is willing to tolerate in the answer. Adding noise reduces the price. Thus, the buyer's query is a pair $\mathbf{Q} = (\mathbf{q}, \nu)$, where \mathbf{q} is a linear query and $\nu \geq 0$ represents an upper bound on the variance, and the price depends on both, $\pi(\mathbf{Q}) = \pi(\mathbf{q}, \nu) \geq 0$. The market maker answers by first computing the exact answer $\mathbf{q}(\mathbf{x})$, then adding noise sampled from a distribution with mean 0 and variance at most ν . This feature gives the buyer more pricing options because, by increasing ν , he can lower his price. Notice that the price depends only on the variance ν , and not on the type of noise.

In the contract with the data owners, the market maker promises to compensate her with a micro-payment. We denote $\mu_i(\varepsilon)$ the sum of all micropayments to all users whose attributes match the i 'th entry to the vector, where ε is the privacy loss incurred by answering the query \mathbf{Q} . The privacy loss depends both on the variance, and on the type of noise that the market maker uses to answer queries: in Section 4

we will restrict the noise to the Laplace distribution, for which there exists an explicit formula connecting the privacy loss ε to the variance. In that case the micro-payment depends on the buyer's query, $\mu_i(\mathbf{Q})$. For the pricing framework to be balanced, we must have $\sum_{i=1}^n \mu_i(\mathbf{Q}) \leq \pi(\mathbf{Q})$. Note that $\mu_i(\mathbf{Q})$ needs to be further split among all users participating in the i 'th bucket of the data \mathbf{x} .

EXAMPLE 3. Continuing Example 2, suppose that there are 1000 voters, and that Bob, the buyer, wants to compute the sum of ratings for candidate A . Assume that each voter requires \$10 for each raw vote. For an accurate answer to the query, Bob needs to pay \$10,000, which is, arguably, too expensive.

Assume Bob is willing to buy the query perturbed with variance $\nu = 5,000$, which would give an error of ± 300 with 94% confidence. The market maker should charge Bob a much lower price; to see how low, we need to consider how the market maker compensates the data owners. We assume that he uses Laplace noise for the perturbation, and therefore one can show that the answer to the query is ε -differentially private with $\varepsilon = 0.1$, which offers reasonably good privacy to all data owners: each will be happy to accept only \$0.001 for basically no loss of privacy, and Bob pays only \$1 for the entire query. The challenge is to design the prices in between. For example, suppose the data owner wants to buy more accuracy, say a variance $\nu = 50$, to reduce the error to ± 30 . How should the price be set? For now, let us observe that the price cannot exceed \$100. If it did, then a savvy buyer would never pay that price, instead he would purchase the \$1 query 100 times, compute the average, and obtain the answer with a variance of $5000/100 = 50$. This is an example of arbitrage and the market maker should define a pricing function that avoids it.

While in the contract with the buyer the price depends only on the variance and not on the type of perturbation, the contract with the data owner is highly sensitive to the type of noise. For example, consider noise defined by the following probability distribution: $P(0) = 1 - 2/m$ and $P(\pm m) = 1/m$, where $m = 10^{64}$. This noise distribution has mean 0 and variance $2m$, but is a poor choice for this market. On the one hand, it has a high variance, which implies a low price π . On the other hand, it returns an accurate answer with extremely high probability, leading to huge privacy losses ε_i , and, consequently, to huge micro-payments. The market maker will not be able to recover his costs. Thus, in order to design a balanced pricing framework, we need to have a perturbation mechanism where the privacy loss is given by an explicit function in the variance; in Section 5.1 we will only consider the Laplace mechanism, where such a function exists.

3. PRICING QUERIES

In this section we describe the first component of the framework as shown in Figure 1: the pricing function $\pi(\mathbf{Q}) = \pi(\mathbf{q}, \nu)$. We denote $\mathbb{R}^+ = [0, \infty)$ and $\bar{\mathbb{R}}^+ = \mathbb{R}^+ \cup \{\infty\}$.

DEFINITION 4. A pricing function is $\pi : \mathbb{R}^n \times \bar{\mathbb{R}}^+ \rightarrow \bar{\mathbb{R}}^+$.

In our framework, the buyer is allowed to issue multiple queries. As a consequence, an important concern is that the buyer may combine answers from multiple queries and

derive an answer to a new query, without paying the full price for the latter, a situation we call *arbitrage*. A reasonable pricing function must guarantee that no arbitrage is possible, in which case we call it *arbitrage-free*. Such a pricing function ensures that the market maker receives proper payment for each query by removing any incentive for the buyer to “game” the system by asking a set of cheaper queries in order to obtain the desired answer. Here we formally define arbitrage-free pricing functions, study their properties, and discuss the construction of arbitrage-free pricing functions.

3.1. Queries and answers

A *randomized mechanism* is a random function \mathcal{K} , with some input \mathbf{x} , denoted $\mathcal{K}(\mathbf{x})$. For a given query $\mathbf{Q} = (\mathbf{q}, v)$, the market maker answers it using a randomized mechanism $\mathcal{K}_{\mathbf{Q}}$, with the property that, for any \mathbf{x} , $\mathbf{E}(\mathcal{K}_{\mathbf{Q}}(\mathbf{x})) = \mathbf{q}(\mathbf{x})$ and $\text{Var}(\mathcal{K}_{\mathbf{Q}}(\mathbf{x})) \leq v$. In other words, when the buyer asks for a query \mathbf{Q} , the market maker samples one value from the distribution $\mathcal{K}_{\mathbf{Q}}$ and returns it to the buyer, in exchange for payment $\pi(\mathbf{Q})$. We abbreviate $\mathcal{K}_{\mathbf{Q}}$ with \mathcal{K} when \mathbf{Q} is clear from the context.

DEFINITION 5. We say that a random function $\mathcal{K}(\mathbf{x})$ answers the query $\mathbf{Q} = (\mathbf{q}, v)$ on the database \mathbf{x} if its expectation is $\mathbf{q}(\mathbf{x})$ and its variance is less than or equal to v .

Other options for answering queries are possible, and we briefly discuss them in Section 6.

To illustrate with a simple example, consider the mechanism $\mathcal{K}_{\mathbf{Q}}$ which, on input \mathbf{x} first computes the query $\mathbf{q}(\mathbf{x})$, then adds random noise with mean 0 and variance v . In this section we do not impose any restrictions on the type of perturbation used in answering the query.

We assume that the market maker is stateless: he does not keep a log of previous users, their queries, or of released answers. As a consequence, each query is answered using an independent random variable. If the same buyer issues the same query repeatedly, the market maker answers using independent samples from the random function \mathcal{K} . Of course, the buyer would have to pay for each query separately.

3.2. Answerability

Before investigating arbitrage we establish the key concept of query answerability. This notion is well studied for deterministic queries and views,^{14,22} but, in our setting, the query answers are random variables, and it requires a precise definition.

DEFINITION 6 (ANSWERABILITY). A query \mathbf{Q} is answerable from a multi-set of queries $\mathbf{S} = \{\mathbf{Q}_1, \dots, \mathbf{Q}_k\}$ if there exists a function $f: \mathbb{R}^k \rightarrow \mathbb{R}$ such that, for any mechanisms $\mathcal{K}_1, \dots, \mathcal{K}_k$, that answer the queries $\mathbf{Q}_1, \dots, \mathbf{Q}_k$, the composite mechanism defined as $\mathcal{K}(\mathbf{x}) \stackrel{\text{def}}{=} f(\mathcal{K}_1(\mathbf{x}), \dots, \mathcal{K}_k(\mathbf{x}))$ answers the query \mathbf{Q} .

We say that \mathbf{Q} is linearly answerable from \mathbf{S} , and write $\mathbf{S} \rightarrow \mathbf{Q}$, if the function f is linear; in that case, we will also say that \mathbf{S} determines \mathbf{Q} .

In other words, if we have already computed some queries, then we can compute a new query by applying a function to their results. For example, suppose we have computed $\mathbf{Q}_1 = (\mathbf{q}_1, v_1)$ and $\mathbf{Q}_2 = (\mathbf{q}_2, v_2)$, and obtained the answers y_1, y_2 .

Then we can answer the query $\mathbf{Q}_3 = ((\mathbf{q}_1 + \mathbf{q}_2)/2, (v_1 + v_2)/4)$ by applying the function $f(y_1, y_2) = (y_1 + y_2)/2$. We pay only for $\mathbf{Q}_1, \mathbf{Q}_2$, and do not pay again for \mathbf{Q}_3 .

How do we check if a query can be answered from a given set of queries? In this paper we give a partial answer, by characterizing when a query is *linearly* answerable.

PROPOSITION 7. Let $\mathbf{S} = \{(\mathbf{q}_1, v_1), \dots, (\mathbf{q}_m, v_m)\}$ be a multi-set of queries, and $\mathbf{Q} = (\mathbf{q}, v)$ be a query. Then the following conditions are equivalent.

1. $\mathbf{S} \rightarrow \mathbf{Q}$.
2. There exists c_1, \dots, c_m such that $c_1 \mathbf{q}_1 + \dots + c_m \mathbf{q}_m = \mathbf{q}$ and $c_1^2 v_1 + \dots + c_m^2 v_m \leq v$.

It follows that deciding determinacy can be done in polynomial time, by solving for the coefficients c_1, \dots, c_m using a quadratic program.

3.3. Arbitrage-free pricing functions: Definition

Arbitrage is possible when the answer to a query \mathbf{Q} can be obtained more cheaply than the advertised price $\pi(\mathbf{Q})$ from an alternative set of priced queries. When arbitrage is possible it complicates the interface between the buyer and market maker: the buyer may need to reason carefully about his queries to achieve the lowest price, while at the same time the market maker may not achieve the revenue intended by some of his advertised prices. Thus, arbitrage is undesirable, and we want to design pricing functions that are arbitrage-free.

DEFINITION 8 (ARBITRAGE-FREE). A pricing function π is arbitrage-free if, whenever a multi-set of queries determines a query, the multi-set is at least as expensive as the query. That is, $\mathbf{S} \rightarrow \mathbf{Q}$ implies $\sum_{\mathbf{Q}' \in \mathbf{S}} \pi(\mathbf{Q}') \geq \pi(\mathbf{Q})$.

If π does permit arbitrage, then a buyer would never pay full price for the determined query, but instead would purchase the multiset of queries that determine it.

EXAMPLE 9. Consider a query (\mathbf{q}, v) offered for price $\pi(\mathbf{q}, v)$. A buyer who wishes to improve the accuracy of the query may ask the same query n times, $(\mathbf{q}, v), (\mathbf{q}, v), \dots, (\mathbf{q}, v)$, at a total cost of $n \cdot \pi(\mathbf{q}, v)$. The buyer then computes the average of the query answers to get an estimated answer with a much lower variance, namely v/n . The pricing function must ensure that the total payment collected from the buyer covers the cost of this lower variance, in other words $n \cdot \pi(\mathbf{q}, v) \geq \pi(\mathbf{q}, v/n)$. If π is arbitrage free, then it is easy to check that this condition holds. Indeed, $\{(\mathbf{q}, v), \dots, (\mathbf{q}, v)\} \rightarrow (n\mathbf{q}, nv) \rightarrow (\mathbf{q}, v/n)$, and arbitrage-freeness implies $\pi(\mathbf{q}, v/n) \leq \pi(\mathbf{q}, v) + \dots + \pi(\mathbf{q}, v) = n \cdot \pi(\mathbf{q}, v)$.

One can prove²⁰ that any arbitrage-free pricing function π has the following properties:

- (1) The zero query is free: $\pi(\mathbf{0}, v) = 0$.
- (2) Higher variance is cheaper: $v \leq v'$ implies $\pi(\mathbf{q}, v) \geq \pi(\mathbf{q}, v')$.

- (3) The zero-variance query is the most expensive^a: $\pi(\mathbf{q}, 0) \geq \pi(\mathbf{q}, \nu)$ for all $\nu \geq 0$.
- (4) Infinite noise is free: if π is continuous at $\mathbf{q} = 0$, then $\pi(\mathbf{q}, \infty) = 0$.
- (5) As $\nu \rightarrow \infty$, $\pi(\mathbf{q}, \nu) = \Omega(1/\nu)$.

Arbitrage-free pricing functions have been studied before,^{17,21} but only in the context of deterministic (i.e. unperturbed) query answers. Our definition extends those in^{17,21} to queries with perturbed answers.

3.4. Arbitrage-free pricing functions: Synthesis

Does an arbitrage-free pricing function exist? The zero-cost function $\pi = 0$, where every query is free, is arbitrage-free. A constant-price pricing function, which charges the same amount $c > 0$ for every query, is not arbitrage-free, because in that case the zero-query would also cost c , and by item (1) above π must have arbitrage. In this section we prove that non-trivial arbitrage-free pricing functions exist, and give some sufficient rules for synthesizing such functions, and in Section 3.5 we show that general construction of arbitrage-free pricing functions remains a difficult problem.

Example 9 and item (5) suggest defining a pricing function that decreases linearly with the variance, $\pi \sim 1/\nu$. Recall that a *semi-norm* is a function $f: \mathbb{R}^n \rightarrow \mathbb{R}^+$ that satisfies the following properties:

- For any $c \in \mathbb{R}$ and any $\mathbf{q} \in \mathbb{R}^n$, $f(c\mathbf{q}) = |c|f(\mathbf{q})$.
- For any $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^n$, $f(\mathbf{q}_1 + \mathbf{q}_2) \leq f(\mathbf{q}_1) + f(\mathbf{q}_2)$.

One can check that, $\mathbf{q} = 0$ implies $f(\mathbf{q}) = 0$; if the converse also holds, then f is called a norm. We prove:

PROPOSITION 10 (BASIC PRICING FUNCTIONS). *Let π be a pricing function of the form $\pi(\mathbf{q}, \nu) = f^2(\mathbf{q})/\nu$, for some function f . Then π is arbitrage-free iff f is a semi-norm.*

Thus, we can obtain an arbitrage-free pricing function from a semi-norm. However, all these pricing functions set an infinite price for the raw (unperturbed) data, because in that case the variance is $\nu = 0$. In some cases, the market maker may be willing to sell the raw data for some high, but finite price. We describe next a method for synthesizing other arbitrage-free pricing functions, including ones that set a finite price for the raw data.

PROPOSITION 11 (COMPOSED PRICING FUNCTIONS). *Let $f: (\mathbb{R}^+)^k \rightarrow \mathbb{R}^+$ be a function that is sub-additive and non-decreasing (meaning $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$, and $\mathbf{x} \leq \mathbf{y}$ implies $f(\mathbf{x}) \leq f(\mathbf{y})$). Then, for any arbitrage-free pricing functions π_1, \dots, π_k the function $\pi = f(\pi_1, \dots, \pi_k)$ is also arbitrage-free.*

A simple sufficient criterion for checking if a function f satisfies the properties required by the proposition is the following: if f is continuous, twice differentiable, $f(\mathbf{0}) = 0$, $\partial f / \partial x_i \geq 0$, $\partial^2 f / \partial x_i \partial x_j \leq 0$ for all $i, j = 1, k$, then it is non-decreasing and sub-additive.

^a It is possible that $\pi(\mathbf{q}, 0) = \infty$.

By using these two propositions we can construct a large set of arbitrage-free pricing functions. We list here a few:

COROLLARY 12. *All functions defined below are arbitrage-free, assuming π_1, \dots, π_k are arbitrage free:*

| | |
|------------------------------------|--|
| <i>L_2-norm:</i> | $\pi(\mathbf{q}, \nu) = \ \mathbf{q}\ _2^2 / \nu = \sum_i q_i^2 / \nu$ |
| <i>L_∞-norm:</i> | $\pi(\mathbf{q}, \nu) = \ \mathbf{q}\ _\infty^2 / \nu = \max_i q_i^2 / \nu$ |
| <i>Weighted L_2:</i> | $\pi(\mathbf{q}, \nu) = (\sum_i w_i \cdot q_i^2) / \nu$ where $w_1, \dots, w_n \geq 0$ |
| <i>Linear comb.:</i> | $c_1 \pi_1 + \dots + c_k \pi_k$ where $c_1, \dots, c_k \geq 0$ |
| <i>Maximum:</i> | $\max(\pi_1, \dots, \pi_k)$ |
| <i>Cut-off:</i> | $\min(\pi_1, c)$ where $c \geq 0$ |
| <i>Power:</i> | π_1^c where $0 < c \leq 1$ |
| <i>Logarithmic:</i> | $\log(\pi_1 + 1)$ |
| <i>Geometric mean:</i> | $\sqrt{\pi_1 \cdot \pi_2}$ |
| <i>Bounded f:</i> | $f(\pi_1)$ where $f \in \{\text{atan}, \text{tanh}, x / \sqrt{x^2 + 1}\}$ |

The last entry defines bounded pricing functions, in particular they charge a finite price for the raw data.

3.5. Deriving pricing functions from price points

We discuss here a more flexible framework for defining an arbitrage-free pricing function: the market maker sets the price for a finite set of queries called *views*, then the system automatically extrapolates this to a pricing function on all queries.

DEFINITION 13. *A price point is a pair (\mathbf{V}, p) , where $\mathbf{V} = (\mathbf{q}, \nu)$ is a query (called a view) and $p \in \mathbb{R}^+$ is a fixed price.*

Given a set of price points $\mathcal{V} = ((\mathbf{V}_1, p_1), \dots, (\mathbf{V}_m, p_m))$, we say that a pricing function π is valid w.r.t. \mathcal{V} if it is arbitrage-free and agrees with all price points, $\pi(\mathbf{V}_i) = p_i$, for all $(\mathbf{V}_i, p_i) \in \mathcal{V}$,

This represents an ideal pricing framework because the market maker only needs to set prices for a small number of queries and variances, and the system extrapolates it to a valid pricing function. However, the technical challenge is: how do we compute a valid pricing function from a set of price points?

In general, there may not exist a valid pricing function. For example, the market maker may choose two views such that the first determines the second, yet the price of the second is set higher than that of the first view. No arbitrage-free pricing function can agree with both price points. If any valid pricing function exists, then we say that the set of price points is *consistent*.

The key to designing a valid pricing function is to compute, for any given query, the cheapest plan to answer it from the views in the set of price points. A *procurement plan* is a plan for answering a query by purchasing views in the set of price points.

DEFINITION 14 (PROCUREMENT PLAN). *Consider an ordered set of price points, $\mathcal{V} = \{(\mathbf{V}_1, p_1), (\mathbf{V}_2, p_2), \dots, (\mathbf{V}_m, p_m)\}$. A procurement plan is an ordered multi-set of non-negative integers, $\mathbf{B} = \{b_1, b_2, \dots, b_m\}$, such that the multiset $\mathcal{V}^{\mathbf{B}} = \{\mathbf{V}_1^{b_1}, \dots, \mathbf{V}_m^{b_m}\}$ (where each query \mathbf{V}_i occurs b_i times) determines \mathbf{Q} : $\mathcal{V}^{\mathbf{B}} \rightarrow \mathbf{Q}$. The cost of the procurement plan is $\text{cost}(\mathbf{B}) = \sum_i b_i p_i$.*

For a simple example, suppose there is a single price point, $\mathcal{V} = \{((\mathbf{q}, 100), \$5)\}$, which charges \$5 for some query with variance 100. The buyer wants to purchase $\mathbf{Q} = (\mathbf{q}, 25)$. Then, a procurement plan is $\{b_1 = 4\}$. In other words, we must purchase the query 4 times, and compute the average, in order to reduce the variance to 25 (since the random noise added to different purchases is independent).

Procurement plans should not be confused with answerability (Definition 6). There the queries were already purchased. Now we have to decide which views to purchase and, in particular, we may purchase a view several times. We call the cost of the cheapest procurement plan the *arbitrage pricing function*.

DEFINITION 15. *Given a set of price-points \mathcal{V} , the arbitrage pricing function is:*

$$\pi_{\mathcal{V}}(\mathbf{Q}) = \min\{\text{cost}(\mathbf{B}) \mid \mathcal{V}^{\mathbf{B}} \rightarrow \mathbf{Q}\}$$

Note that, by this definition, if a query is not answerable from a set of price points, then the arbitrage price is ∞ . The arbitrage function has a simple interpretation. Once the views are offered for sale, a buyer can purchase any query and pay only the arbitrage function, regardless of what the market maker wants to charge for that query. The arbitrage function is thus an upper bound on any valid pricing function. Could the market maker choose the arbitrage function as the official price for all queries? We answer this positively, by proving that the arbitrage function is indeed arbitrage-free.

THEOREM 16. *For any set of price points \mathcal{V} , the arbitrage function $\pi_{\mathcal{V}}$ is arbitrage-free. \mathcal{V} is consistent, if the arbitrage price does not lower the price of any view: $p_i \leq \pi_{\mathcal{V}}(V_i)$ for all $(V_i, p_i) \in \mathcal{V}$.*

Thus, one choice for the market maker is to set all prices to be the arbitrage function. Unfortunately, it turns out that computing the arbitrage function is NP-hard.

THEOREM 17. *The following problem: “Given a set of price points \mathcal{V} , a query \mathbf{Q} , and budget $p > 0$, decide whether \mathbf{Q} is answerable from \mathcal{V} within a budget p (in other words, check $\pi_{\mathcal{V}}(\mathbf{Q}) \leq p$)”, is NP-complete.*

The proof is by reduction from the *Exact Cover by 3-Sets* problem. Hardness holds even if all views in \mathcal{V} have variance 0.

4. PRIVACY LOSS

In this section we describe the second component of the pricing framework as shown in Figure 1: the privacy loss for each owner, denoted by ε . Our definition of privacy loss follows that of differential privacy. Given the database vector \mathbf{x} , denote by $\mathbf{x}^{(j)}$ the database vector that results from adding one count to entry j and leaving all other values unchanged. An individual’s privacy loss is measured by comparing the mechanism output on two data vectors that differ in *any* one entry.

DEFINITION 18. *Let \mathcal{K} be any mechanism (meaning: for any database instance \mathbf{x} , $\mathcal{K}(\mathbf{x})$ is a random variable). The privacy loss to each user, in notation $\varepsilon(\mathcal{K}) \in \mathbb{R}^+$, is defined as:*

$$\varepsilon(\mathcal{K}) = \sup_{S, x, j} \left| \log \frac{\Pr(\mathcal{K}(\mathbf{x}) \in S)}{\Pr(\mathcal{K}(\mathbf{x}^{(j)}) \in S)} \right|$$

where \mathbf{x} ranges over all possible databases and S ranges over measurable sets of \mathbb{R} .

Recall that in Section 3 we defined the price $\pi(\mathbf{Q})$ to depend only on the variance, and not on the mechanism used to answer the query. But the definition of privacy loss depends on the mechanism \mathcal{K} , so now we need to choose a particular mechanism. We will use the classical Laplace mechanism; in that case, the privacy loss can be given as a function of the variance, and of a property that depends only on the query, called *query sensitivity*.⁷ The sensitivity of \mathbf{q} , denoted $s_{\mathbf{q}}$, is the largest possible change to the answer that can be caused by adding or removing one individual’s contribution: $s_{\mathbf{q}} \stackrel{\text{def}}{=} \sup_{x, j} |\mathbf{q}(\mathbf{x}) - \mathbf{q}(\mathbf{x}^{(j)})|$.

Formally, the *Laplace Mechanism*, denoted \mathcal{L} , answers a query $\mathbf{Q} = (\mathbf{q}, v)$ by returning $\mathcal{L}_{\mathbf{Q}}(\mathbf{x}) = \mathbf{q}(\mathbf{x}) + \rho$, where ρ is noise drawn from a Laplace distribution, centered at zero, with scale $b = \sqrt{v/2}$. The privacy loss of each individual is bounded by $\varepsilon(\mathcal{L}_{\mathbf{Q}}) \leq \frac{s_{\mathbf{q}}}{\sqrt{v/2}}$.⁶

5. BALANCED PRICING FRAMEWORKS

In this section we define formally when a pricing framework is *balanced* and we provide a general procedure for designing a balanced pricing framework. The concept of balance brings together the three components as shown in Figure 1: the query price π , the privacy loss ε , and the micro-payments μ_i . We begin with a description of micro-payments.

5.1. Micro-payments to data owners

By answering a buyer’s query \mathbf{Q} , using some mechanism $\mathcal{K}_{\mathbf{Q}}$, the market maker leaks some of the private data of the data owners. He must compensate each data owner with some micro-payment. Recall that $\mu_i(\mathbf{Q})$ denotes the sum of all micro-payments due to the data owner whose attributes match x_i , the i ’th entry in the data vector. The micro-payments close the loop as shown in Figure 1: they must be covered by the buyer’s payment π , and must also be a function of the degree of the privacy loss ε . We require the micro-payment functions μ_i to be arbitrage-free, in order to promise that the owner’s loss of privacy will be compensated, and that there is no way for the buyer to circumvent the owed micro-payment by asking other queries and combining their answers.

5.2. Balanced pricing frameworks: Definition

The contract between data owners and the market-maker consists of non-decreasing functions $W_i: \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $i = 1, n$, s.t. $W_i(0) = 0$. Here $\mathbb{R}^+ = \mathbb{R}^+ \cup \{+\infty\}$: the market maker promises to pay the data owners contributing to the i ’ bucket a micro-payment of at least $\mu_i \geq W_i(\varepsilon)$ in the event of a privacy loss ε . We denote $\mathbf{W} = (W_1, \dots, W_n)$ the set of contracts between the market maker and all data owners.

The connection between the micro-payments μ_i , the query price π and the privacy loss ε_i is captured by the following definition.

DEFINITION 19. We say that the micro-payment functions μ_i , $i = 1, \dots, n$ are cost-recovering for a pricing function π if, for any query \mathbf{Q} , $\pi(\mathbf{Q}) \geq \sum_i \mu_i(\mathbf{Q})$.

Fix a query answering mechanism \mathcal{K} . We say that a micro-payment function μ_i is compensating for a contract function W_i if for any query \mathbf{Q} , $\mu_i(\mathbf{Q}) \geq W_i(\varepsilon(\mathcal{K}_Q))$.

The market maker will insist that the micro-payment functions are cost-recovering; otherwise, he will not be able to pay the data owners from the buyer's payment. In addition, a data owner will insist that the micro-payment function is compensating: this enforces the contract between her and the market maker, guaranteeing that she will be compensated at least $W_i(\varepsilon)$, in the event of a privacy loss ε .

Fix a query answering mechanism \mathcal{K} . We denote a pricing framework $(\pi, \varepsilon, \mu, \mathbf{W})$, where $\pi(\mathbf{Q})$, $\mu_i(\mathbf{Q})$ are the buyer's price and the micro-payments, $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$ where $\varepsilon_i(\mathcal{K}_Q)$ is the privacy loss corresponding to the mechanism \mathcal{K} , and $W_i(\varepsilon)$ is the contract with the data owner i .

DEFINITION 20. A pricing framework $(\pi, \varepsilon, \mu, \mathbf{W})$ is balanced if (1) π is arbitrage-free and (2) the micro-payment functions μ are arbitrage-free, cost-recovering for π , and compensating for \mathbf{W} .

We explain how the contract between the data owner and the market maker differs from that in privacy-preserving mechanisms. Let $\varepsilon > 0$ be a small constant. A mechanism \mathcal{K} is called *differentially private*⁷ if, for any measurable set S , any database vector \mathbf{x} and for any entry j of \mathbf{x} :

$$e^{-\varepsilon} \cdot \Pr(\mathcal{K}(\mathbf{x}^{(j)}) \in S) \leq \Pr(\mathcal{K}(\mathbf{x}) \in S) \leq e^\varepsilon \cdot \Pr(\mathcal{K}(\mathbf{x}^{(j)}) \in S)$$

In differential privacy, the basic contract between the mechanism and the data owner is the promise to every user that her privacy loss is no larger than ε . In our framework for pricing private data we turn this contract around. Now, privacy is lost, and Definition 18 quantifies this loss. The contract requires that the users are compensated according to their privacy loss. At an extreme, if the mechanism is ε -differentially private for a tiny ε , then each user will receive only a tiny micro-payment $W_i(\varepsilon)$; as her privacy loss increases, she will be compensated more.

The micro-payments circumvent a common limitation of differentially-private mechanisms. In differential privacy, the data analyst typically has a fixed budget, ε , granted by the data curator, for all queries that he may ever ask. In order to issue N queries, he needs to divide the privacy budget among these queries, and, as a result, each query will be perturbed with greater noise. After issuing these N queries, he can no longer query the database, because otherwise the contract with the data owner would be breached.

In our pricing framework there is no such hard limitation because the buyer simply pays for each query. The budget is now a real financial quantity, and the buyer can ask as many query as he wants, with as high accuracy as he wants, as long as he has money to pay for them. As a consequence, it is the analyst-buyer, rather than the data owner, who ultimately determines the actual privacy loss. A similar model

was proposed by Ghosh et al., in which the ε budget is determined by the budget of the data analyst.¹²

5.3. Balanced pricing frameworks: Synthesis

Call $(\varepsilon, \mu, \mathbf{W})$ *semi-balanced* if all micro-payment functions are arbitrage free and compensating w.r.t. \mathcal{K} ; that is, we leave out the pricing function π and the cost-recovering requirement. The first step is to design a semi-balanced set of micro-payment functions.

PROPOSITION 21. Let \mathcal{L} be the Laplace Mechanism, and let the contract functions be linear, $W_i(\varepsilon) = c_i \cdot \varepsilon$, where $c_i > 0$ is a fixed constant, for $i = 1, \dots, n$. Define the micro-payment functions $\mu_i(\mathbf{Q}) = \frac{\max_j |q_j| \cdot c_i}{\sqrt{v/2}}$, for $i = 1, \dots, n$. Then $(\varepsilon, \mu, \mathbf{W})$ is semi-balanced.

The proposition defines the micro-payment associated to linear contracts W_i , which requires infinite payment for total loss of privacy. In practice, data owners are willing to sell their raw data at some high (but finite) price. We show next how to derive new semi-balanced micro-payments by applying sub-additive functions to other micro-payments.

PROPOSITION 22. Fix k semi-balanced pricing frameworks, $(\varepsilon, \mu^j, \mathbf{W}^j)$, $j = 1, \dots, k$. Define $\mu_i = f_i(\mu_i^1, \dots, \mu_i^k)$, and $W_i = f_i(W_i^1, \dots, W_i^k)$, for each $i = 1, \dots, n$, where each function $f_i: (\mathbb{R}^+)^k \rightarrow \mathbb{R}^+$, $i = 1, \dots, n$, is non-decreasing, sub-additive, and satisfies $f_i(\mathbf{0}) = 0$. Then, $(\varepsilon, \mu, \mathbf{W})$ is also semi-balanced, where $\mu = (\mu_1, \dots, \mu_n)$ and $\mathbf{W} = (W_1, \dots, W_n)$.

Finally, we choose a payment function such as to ensure that the micro-payments are cost-recovering.

PROPOSITION 23. Suppose that $(\varepsilon, \mu, \mathbf{W})$ is semi-balanced, and define $\pi(\mathbf{Q}) = \sum_i \mu_i(\mathbf{Q})$. Then, $(\pi, \varepsilon, \mu, \mathbf{W})$ is balanced.

To summarize, the synthesis procedure for a pricing framework proceeds as follows. Start with the simple micro-payment functions given by Proposition 21, which ensure linear compensation for each user. Next, modify both the micro-payment and the contract functions using Proposition 22, as desired, in order to adjust to the preferences of individual users (e.g. in order to allow a user to set a price for her true data). Finally, define the query price to be the sum of all micro-payments (Proposition 23), then increase this price freely, by using any method in Section 3.4.

6. DISCUSSION

A number of issues deserve further attention if we are to achieve a fully-functional marketplace for private data.

First, we have made two restrictions which limit the prevention of arbitrage in our market. We require the market maker to answer queries using an unbiased estimator (Definition 5) and therefore arbitrage-freeness only prevents an adversary from deriving an unbiased estimator of a query using cheaper queries (Definition 6). In addition, we have considered a restricted notion of answerability, limiting our attention to adversaries who only consider queries computed by linear functions of other queries. Both of these

assumptions limit the class of adversaries against which the arbitrage-free property is guaranteed to hold.

A more general approach would be to relax Definition 5 to allow queries to be answered using either biased or unbiased estimators and to include answerability using non-linear functions. This would provide the market maker with a stronger guarantee against arbitrage. However it would likely make reasoning about determinacy and arbitrage-free pricing significantly more difficult, and it would further restrict the set of arbitrage-free pricing functions available to the market maker. In other words, a more powerful notion of arbitrage would lead to more restrictive pricing functions, potentially limiting the ability of the market maker to set prices. The tradeoff between completeness of the framework and the feasibility of analyzing arbitrage is an important topic for future research.

A second issue is the problem of incentivizing the data owner to participate in the database and truthfully report her privacy valuations. Currently, there is nothing stopping the data owner from quoting an impossibly high price, for even a tiny loss of her privacy. In other words, she would choose a contract function $W(\varepsilon)$ that is as close to ∞ as possible. Incentivizing users to report their true valuation is a goal of *mechanism design*. This has been studied for private data only in the restricted case of a single query, and has been shown to be a difficult task.^{10, 12} In Ref. Li et al.²⁰ we make a preliminary attempt to address this issue by requiring that users choose from a limited set of contract functions (e.g. one appropriate for a risk-tolerant user and one appropriate for a risk-averse user).

A third issue is the protection of privacy valuations themselves. When a user has sufficient freedom to choose their privacy valuation, it may be strongly correlated with the data x_i itself. In that case, even releasing the price of a query may lead to privacy loss, a factor not considered in the framework described above. Hiding the valuation itself is a difficult problem which is still being actively investigated in mechanism design.^{10, 12, 23} In, Ref. Li et al.²⁰ we describe a simple approach that is based on perturbing the price itself, in the same way that we perturb the data. Thus both $\pi(Q)$ and $\mu_i(Q)$ are perturbed in the same fashion as query answers, and are therefore random variables. All three properties of arbitrage-freeness, cost-recovery, and compensation are then defined in terms of expected values. In addition, the privacy loss for data item x_i includes two parts: one is due to the release of the query answer and the other is due to the release of the price.

7. CONCLUSION

We have introduced a framework for selling private data. Buyers can purchase any linear query, with any amount of perturbation, and need to pay accordingly. Data owners, in turn, are compensated according to the privacy loss they incur for each query. Buyers are allowed to ask an arbitrary number of queries, and we have designed techniques for ensuring that the prices are arbitrage-free, according to a specific definition of arbitrage-freeness, meaning that buyers are guaranteed to pay for any information they may further extract from the queries. Our pricing framework is balanced, in the sense that the buyer's price covers the

micro-payments to the data owner and each micro-payment compensates the users according to their privacy loss.

An interesting open question is whether we can achieve both truthfulness (e.g. as discussed in Ghosh and Roth¹²) and arbitrage-freeness (as discussed in the current paper) when pricing private data. Further, it remains to consider general notions of answerability that go beyond linear answerability, or to bound the impact non-linear estimation methods could have in the context of arbitrage. **□**

References

- Acquisti, A., Taylor, C.R., Wagman, L. The economics of privacy. *Journal of Economic Literature* 52, 2 (2016).
- Balazinska, M., Howe, B., Suci, D. Data markets in the cloud: An opportunity for the database community. *PVLDB* 4, 12 (2011), 1482–1485.
- Calo, R. The boundaries of privacy harm. *Indiana Law Journal* 86, 3 (2011).
- Chen, B.-C., Kifer, D., LeFevre, K., Machanavajjhala, A. Privacy-preserving data publishing. In *Foundations and Trends in Databases* (2010).
- Dinur, I., Nissim, K. Revealing information while preserving privacy. In *PODS* (2003), 202–210.
- Dwork, C. A firm foundation for private data analysis. *Commun. ACM* 54, 1 (2011), 86–95.
- Dwork, C., McSherry, F., Nissim, K., Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006), 265–284.
- Dwork, C., Roth, A. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science (2014) Now Publishers Inc., Hanover, MA, USA.
- Erlingsson, U., Pihur, V., Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Computer and Communications Security (CCS)* (2014), 1054–1067.
- Fleischer, L., Lyu, Y.-H. Approximately optimal auctions for selling privacy when costs are correlated with data. In *ACM Conference on Electronic Commerce* (2012), 568–585.
- Fung, B.C.M., Wang, K., Chen, R., Yu, P.S. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.* 42, 4 (2010).
- Ghosh, A., Roth, A. Selling privacy at auction. In *ACM Conference on Electronic Commerce* (2011), 199–208.
- Greenberg, A. Apple's "differential privacy" is about collecting your data—but not your data. *Wired* (Jun 13 2016).
- Halevy, A.Y. Answering queries using views: A survey. *VLDB J.* 10, 4 (2001), 270–294.
- Hardt, M., Ligett, K., McSherry, F. A simple and practical algorithm for differentially private data release. In *NIPS* (2012).
- Kifer, D., Abowd, J., Gehrke, J., Vithuber, L. Privacy: Theory meets practice on the map. In *ICDE* (2008).
- Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., Suci, D. Query-based data pricing. *J. ACM* 62, 5 (Nov. 2015), 43:1–43:44.
- Laudon, K.C. Markets and privacy. *Commun. ACM* 39, 9 (1996), 92–104.
- Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A. Optimizing linear counting queries under differential privacy. In *PODS* (2010).
- Li, C., Li, D.Y., Miklau, G., Suci, D. A theory of pricing private data. *ACM Trans. Database Syst.* 39, 4 (Dec. 2014), 1–28.
- Li, C., Miklau, G. Pricing aggregate queries in a data marketplace. In *WebDB* (2012).
- Nash, A., Segoufin, L., Vianu, V. Views and queries: Determinacy and rewriting. *TODS* 35, 3 (2010).
- Nissim, K., Vadhan, S., Xiao, D. Redrawing the boundaries on purchasing data from privacy-sensitive individuals. In *Conference on Innovations in Theoretical Computer Science* (2014), 411–422.

Chao Li (chaoli@google.com), Google Inc.
 Gerome Miklau (miklau@cs.umass.edu),
 University of Massachusetts, Amherst, MA.

Daniel Yang Li and Dan Suci
 ({dyl, suci}@cs.washington.edu),
 University of Washington, Seattle, WA.

Permission to make digital or hard copies of all or part 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 the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

©ACM 0001-0782/17/1200 \$15.00.

Technical Perspective

A Simple, Elegant Approach to Non-Numeric Parallelization

By James Larus

STANDARDIZED INTERFACES PLAY AN important role in many industries; for example, a processor's instruction set architecture (ISA) defines the interface between hardware and software. A stable, well-defined interface facilitates independent innovation on both sides of the boundary. Unchanging (or slowly changing) ISAs have meant that new programming languages, compilers, and applications can run on old computers. At the same time, new computers, with improved microarchitectures, can run existing programs faster.

Recently, however, processors have been extended with a wide range of architecturally visible features that change their ISA, for example, support for vector processing, virtual memory, cryptography, and secure execution.

When does it make sense to extend an interface and introduce a new feature that breaks backward compatibility? Clearly, with something as pervasive as an ISA, change is not to be taken lightly. Software that uses a new feature will benefit only from new computers, which initially are rare. This software must also emulate the feature on older machines, so changing the ISA increases the complexity of both hardware and software and raises the cost of testing and error diagnosis.

The following paper proposes a modest hardware extension to support a new parallel execution model for small, non-numeric loops. Under realistic assumptions, HELEX-RC increases the performance of a handful of well-known, non-numeric benchmarks by over a factor of 6. Many programs contain similar loops and might benefit as well. Although the hardware is simple, it requires pervasive changes to the software stack because the feature is tied to a specific execution model. Existing compiled programs will not benefit from HELEX.

The authors, however, present a powerful argument for paying this price. Short, non-numeric loops are very common, but have long been con-

sidered poor candidates for parallel execution. Each of their iterations typically runs for a short amount of time (in the benchmarks, 25 clock cycles on the average, 50% requiring 10 or few clock cycles). Moreover, non-numeric loops commonly execute conditional branches, so their execution behavior is data dependent. Neither consideration is fatal by itself, but these loops also pass values among iterations. It is difficult to analyze these loop-carried dependencies and to generate low-overhead code that respects them.


So, in general, these loops run sequentially. In the era when computer performance doubled every two years, sequential execution was not a pressing problem. But, CPU clock speed stopped increasing over a decade ago, and multicore parallelism became the preferred approach to utilize the additional transistors provided by Moore's Law. But, today, this parallelism is not widely used outside of datacenters. Mobile phones and laptops contain only 2–4 cores.

Much of the software that runs on these platforms is non-numeric and contains loops with short, data-dependent bodies. The authors argue that our inability to parallelize these loops is a fundamental impediment to exploiting parallel hardware. Amdahl's Law relates the parallel speedup of a program to the fraction of its execution that is sequen-

tial. In a program with 50% of its time spent in sequential loops, even with perfect parallelization of the other code, the largest possible speedup is 2. Clearly something must be done with these non-numeric loops.

The paper proposes a simple and elegant approach to non-numeric parallelization. The HELEX compiler identifies the sequential portions of a loop, which will run code tied to the loop-carried dependencies. These portions must execute in sequential order, and values produced by one iteration must be sent to subsequent iterations, even if they are running on another processor. Normal shared-memory communication can cost hundreds of cycles and so are too expensive for fine-grain iterations that may only run for a fraction of that time. Parallel speedup comes from executing the other portions of a loop—those not involved in the loop-carried dependencies—concurrently.

HELEX-RC is a simple hardware extension that reduces or eliminates communication overhead. It introduces a small amount of buffered memory for loop-carried dependencies and synchronization variables. These values are proactively sent to the processors that will execute subsequent iterations, so these values will be in local memory when an iteration starts, thereby ensuring it is not delayed.

HELEX-RC is not a general-purpose hardware feature like a cache or register file. Its design is tightly tied to the HELEX execution model. Clever compiler writers will no doubt find other uses for it, but until then, it is a specialized solution to one problem. But, small improvements are sometimes the key to unlock the value of much larger architectural features such as multicore parallelism. 

HELEX-RC is a simple hardware extension that reduces or eliminates communication overhead.

James Larus (james.larus@epfl.ch) is Dean of the School of Computer and Communications Science, EPFL, Lausanne, Switzerland.

Copyright held by author.

Automatically Accelerating Non-Numerical Programs by Architecture-Compiler Co-Design

By Simone Campanoni, Kevin Brownell, Svilen Kanev, Timothy M. Jones, Gu-Yeon Wei, and David Brooks

Abstract

Because of the high cost of communication between processors, compilers that parallelize loops automatically have been forced to skip a large class of loops that are both critical to performance and rich in latent parallelism. HELIX-RC is a compiler/microprocessor co-design that opens those loops to parallelization by decoupling communication from thread execution in conventional multicore architectures. Simulations of HELIX-RC, applied to a processor with 16 Intel Atom-like cores, show an average of 6.85× performance speedup for six SPEC CINT2000 benchmarks.

1. INTRODUCTION

On a multicore processor, the performance of a program depends largely on how well it exploits parallel threads. Some computing problems are solved by numerical programs that are either inherently parallel or easy to parallelize. Historically, successful parallelization tools have been able to transform the sequential loops of such programs into parallel form, boosting performance significantly. Most software, however, is still sequentially designed and largely non-numerical, with irregular control and data flow. Because manual parallelization of such software is error-prone and time-consuming, automatic parallelization of non-numerical programs remains an important open problem.

The last decade has seen impressive steps toward a solution, but when targeting commodity processors, existing parallelizers still leave much of the latent parallelism in loops unrealized.⁵ The larger loops in a program can be so hard to analyze accurately that apparent dependences often flood communication channels between cores. Smaller loops are more amenable to accurate analysis, and our work shows that there is a lot of parallelism between the iterations of small loops in non-numerical programs represented by SPECint2000 benchmarks.⁴ But even after intense optimization, small loops typically include loop-carried dependences, so their iterations cannot be entirely independent—they must communicate. Because the iterations of a small loop are short (25 clock cycles on average for SPECint2000), their communications are frequent.

On commodity processors, communication relies on the memory system and is reactive, triggered only when one core asks for data from another. The resulting delay is longer than the average duration of an iteration, and it is hard to overlap with computation, especially when the variance of durations is high, as in non-numerical workloads. The

benefits of automatic loop parallelization therefore saturate at small numbers of cores for commodity processors.

Lowering the latency of inter-core communication would help, but it can only go so far, if communication remains reactive. We therefore propose a proactive solution, in which the compiler and an architectural extension called *ring cache* cooperate to overlap communication with computation and lower communication latency. The compiler identifies data that must be shared between cores, and the ring cache circulates that data as soon as it is generated.

To demonstrate this idea, we have developed HELIX-RC, a co-design incorporating a parallelizing compiler and a simulated chip multiprocessor extended with ring cache. The HELIX-RC compiler builds on the original HELIX code parallelizer for commodity multicore processors.⁵ Because it relies on invariants of the code produced by the compiler, ring cache is a lightweight, non-invasive extension of conventional multicore architecture. Because it facilitates proactive, low-latency inter-core communication, ring cache allows HELIX-RC to outperform HELIX by a factor of 3×.

2. OPPORTUNITIES AND CHALLENGES OF SMALL LOOPS

2.1. Opportunities

Prior loop parallelization techniques have avoided selecting loops with small bodies because communication would slow down execution on conventional processors.^{5, 20} On average, such techniques yield only about 60% coverage by parallelized loops for non-numerical programs. Excluding small loops limits overall speedup of such programs to less than 3 times no matter how many cores are available, because by Amdahl's law, coverage dictates the overall speedup of a program through parallelization.

Because the intricacy of control and data flow scales down with code size, small loops are easier than larger ones for a compiler to analyze, which reduces the proportion of data dependences that must be accommodated at run time because of conservative assumptions about possible pointer aliases. As a result, the optimized bodies of small loops yield relatively independent iteration threads.⁵ So there could be

The original paper, "HELIX-RC: An Architecture-Compiler Co-Design for Automatic Parallelization of Irregular Programs," was published in *Proceedings of the International Symposium on Computer Architecture*, June 14–18, 2014, 217–228.

a significant increase in core utilization, and concomitant overall speedup, if the compiler were able to freely select small hot loops for parallelization. Realizing that potential requires understanding the characteristics of such loops and optimizing for them.

2.2. Low latency challenge

To illustrate the need for low-latency communication, Figure 1a plots a cumulative distribution of average iteration execution times on a single Atom-like core (described in our work⁴) for the set of hot loops from SPECint2000 chosen for parallelization by HELIX-RC. The shaded portion of the plot shows that more than half of the loop iterations complete within 25 clock cycles. The figure also shows the measured core-to-core round trip communication latencies for 3 modern multicore processors. Even the shortest of these latencies, 75 cycles for Ivy Bridge, is too heavy a communication penalty for the majority of these short loops.

2.3. Broadcast challenge

Loops within non-numerical programs generate values that are consumed by later iterations, but the compiler cannot know which iterations will use which values. So when the compiler distributes the iterations of a loop to separate cores, shared values that result from loop-carried dependences need to be accessible to any of those cores soon after being generated.

For loops parallelized by HELIX-RC, most communication of shared values is not between cores executing successive loop iterations, which HELIX-RC assigns to adjacent cores. Figure 1b charts the distribution of value communication distances (defined as the undirected distance between the core that produces a value and the first one that consumes it) on a platform with 16 cores organized in a ring. Only 15% of those transfers are between adjacent cores.

Moreover, Figure 1c shows that most (86%) of the loop-carried values in parallelized loops are consumed by multiple cores. Since consumers of shared values are not known at compile time, especially for non-numerical workloads, broadcasting is the most appropriate communication protocol.

It is well known that implementing low-latency broadcast is challenging for a large set of cores. HELIX-RC uses a hardware mechanism that achieves proactive delayed broadcast

of data and signals to all cores in the ring for a loop. Such proactive communication is the cornerstone of the HELIX-RC approach, which allows the communication needed for sharing data between cores to overlap with computation that the cores carry out in parallel.

3. THE HELIX-RC SOLUTION

To run the iterations of small hot loops efficiently in parallel, HELIX-RC replaces communication-on-demand with proactive communication. It decouples value forwarding between threads from value consumption by the receiving thread. It also decouples transmission of synchronizing signals from the code that enforces sequential semantics. Extensions of conventional microprocessor architecture make this decoupling possible. Reliance on compiler-guaranteed machine code properties keeps those architectural extensions simple and efficient.

3.1. Approach

HELIX-RC is a co-design that binds its compiler (*HCCv3*) to a processor architecture enhancement called *ring cache*. When the compiler generates a set of parallel threads to run on separate cores, they are rarely completely independent. While most of each thread's code can run concurrently with other threads, there are segments of that code that must execute in strict sequence across the thread set. We call these *sequential segments*. The main role of the ring cache is to accelerate the communication of values and synchronizing signals needed to implement sequential segments correctly.

The ring cache is a ring network linking *ring nodes*, each of which is attached to a core in the processor. During sequential segments, this ring serves as a distributed first-level cache preceding the private L1 cache (Figure 2). *HCCv3* marks the entry and exit points of sequential segments using 2 instructions that extend the instruction set. As a result, each core knows whether or not it is currently executing the sequential segment of a parallel thread, and it accesses the cache hierarchy accordingly.

Figure 1. Small hot loops have short iterations that send data over multiple hops and to multiple cores.

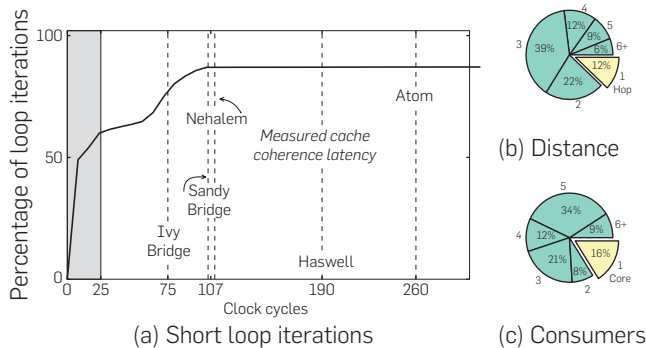
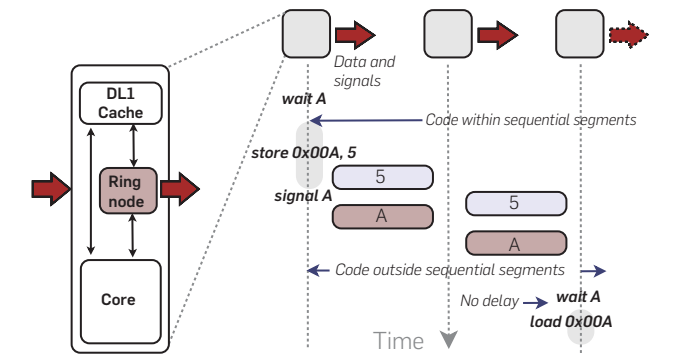


Figure 2. The ring cache is a ring network that connects ring nodes attached to each core. It operates during sequential segments as a distributed first-level cache that precedes the private L1 cache (left side). Ring nodes propagate newly-generated values without involving their attached cores (right side). In this example, data generated by the rightmost core is available at the leftmost core when needed, so wait A incurs no delay.



Compiler. HCCv3 automatically generates parallel threads from sequential programs by distributing successive loop iterations across adjacent cores organized as a unidirectional ring within a single multicore processor. HCCv3 parallelizes loops that are most likely to speed up performance when their iterations execute in parallel. Only 1 loop runs in parallel at a time.

To preserve the sequential semantics of the original loop, the code that implements a loop-carried data dependence, that is, one spanning loop iterations, must run in a sequential segment whose instances in parallel threads execute in iteration order. Variables and other data structures involved in such dependences—even those normally allocated to registers in sequential code—are mapped to specially-allocated memory locations shared between cores. HCCv3 guarantees that accesses to those shared memory locations always occur within sequential segments.

ISA. We introduce a pair of instructions—`wait` and `signal`—that mark the beginning and end of a sequential segment. Each has an integer operand that identifies the particular sequential segment. A `wait 3` instruction, for example, blocks execution of the core that issues it until all other cores running earlier iterations have finished executing the sequential segment labeled 3, which they signify by executing `signal 3`. Figure 2 shows a sequential segment with label A being executed by the core attached to the leftmost ring node. Between `wait A` and `signal A`, a `store` instruction sends the new value 5 for the shared location at address `0x00A` to the ring node for caching and circulation to its successor nodes. The `signal A` instruction that ends the segment also signals subsequent nodes that the value generated by segment A is ready.

A core forwards all memory accesses within sequential segments to its local ring node. All other memory accesses (not within a sequential segment) go through the private L1 cache.

Memory. Each ring node has a cache array that satisfies both loads and stores received from its attached core during a sequential segment. HELIX-RC does not require other changes to the existing memory hierarchy because the ring cache orchestrates interactions with it. To avoid any changes to conventional cache coherence protocols, the ring cache permanently maps each memory address to a unique ring node. All accesses from the distributed ring cache to the next cache level (L1) go through the associated node for a corresponding address.

3.2. Overlapping communication with computation

Because shared values produced by a sequential segment and the signal that marks its end are propagated through the ring node as soon as they are generated, this communication between iterations is decoupled from computation taking place on the cores.

Shared data communication. Once a ring node receives a store, it records the new value and proactively forwards its address and value to an adjacent node in the ring cache, all without interrupting the execution of the attached core. The value then propagates from node to node through the rest of the ring without interrupting the computation of any core.

Synchronization. Given the difficulty of determining which iteration depends on which in non-numerical programs, compilers typically make the conservative assumption that an iteration depends on all of its predecessor iterations. Therefore, a core cannot execute sequential code until it is unblocked by its predecessor.^{5,20} Moreover, an iteration unblocks its successor only if both it and its predecessors have executed this sequential segment or if they are not going to. This execution model leads to a chain of signal propagation across loop iterations that includes unnecessary synchronization: even if an iteration is not going to execute sequential code, it still needs to synchronize with its predecessor before unblocking its successor.

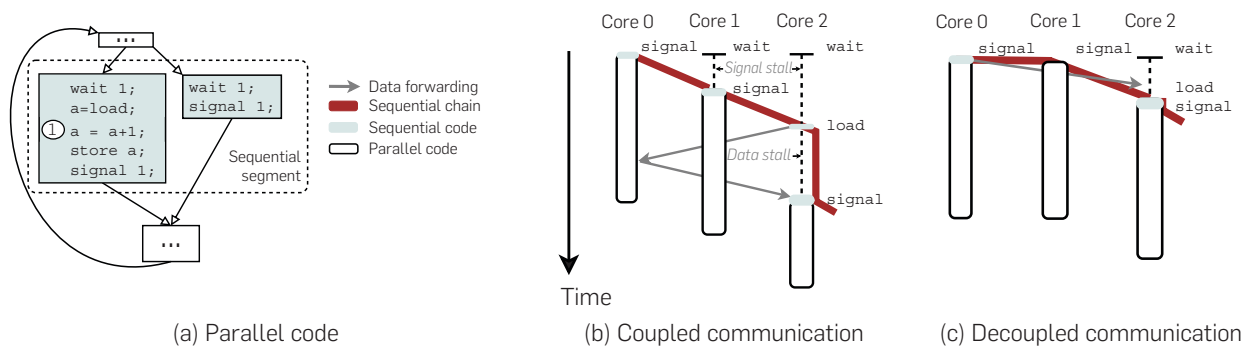
HELIX-RC removes this synchronization overhead by enabling an iteration to detect the readiness of all predecessor iterations, not just one. Therefore, once an iteration forgoes executing the sequential segment, it immediately notifies its successor without waiting for its predecessor. Unfortunately, while HELIX-RC removes unnecessary synchronization, it increases the number of signals that can be in flight simultaneously.

HELIX-RC relies on the `signal` instruction to handle synchronization signals efficiently. Synchronization between a producer core and a consumer includes generation of a signal by the producer, a request for that signal by the consumer, and transmission of the signal between the two. On a conventional multicore processor that relies on a demand-driven memory system for communication, signal transmission is inherently lazy, and signal request and transmission become serialized. With HELIX-RC, on the other hand, a `signal` instructs the ring cache to proactively forward a signal to all other nodes in the ring without interrupting any of the cores, thereby decoupling signal transmission from synchronization.

Code example. The code in Figure 3(a), abstracted for clarity, represents a small hot loop from `175.vpr` of SPEC CINT2000. It is responsible for 55% of the total execution time of that program. The loop body has 2 execution paths. The left path entails a loop-carried data dependence because during a typical loop iteration, instruction 1 uses the value of variable `a` produced by a previous iteration. The right path does not depend on prior data. Owing to complex control flow, the compiler cannot predict the execution path taken during a particular iteration, so it must assume that instruction 1 may depend on the previous iteration.

In a conventional implementation coupling communication with computation, the compiler would add `wait 1` and `signal 1` instructions to the right path, as shown in Figure 3(a), to synchronize each iteration with its predecessor and successor iterations. If shared values and signals were communicated on demand, the resulting sequential signal chain would look like that highlighted in red shown in Figure 3(b). If we assume that only iterations 0 and 2, running on cores 0 and 2, respectively, take the left path and execute instruction 1, then the sequential signal chain shown in Figure 3(b) is unnecessarily long, because iteration 1 only executes parallel code, so the `wait` instruction is unnecessary in that iteration. It results in a signal stall. Iterations 0 and 2, in order to update `a`, must load its

Figure 3. Example illustrating benefits of decoupling communication from computation.



previous value first, using a regular load. So lazy forwarding of this shared data leads to data stalls, because the transfer only begins when demanded by a load, rather than when generated by a store.

In HELIX-RC, however, a `wait` A unblocks when *all* predecessor iterations have signaled that segment A is finished. That allows HCCv3 to omit the `wait 1` on the right path through the loop body. That optimization, combined with HELIX-RC’s proactive communication between cores, leads to the more efficient scenario shown in Figure 3(c). The sequential chain in red now only includes the delay required to satisfy the dependence—communication updating a shared value.

4. COMPILER

The decoupled execution model of HELIX-RC described so far is possible given the tight co-design of the compiler and architecture. In this section, we focus on compiler-guaranteed code properties that enable a lightweight ring cache design, and follow up with code optimizations that make use of the ring cache.

4.1. Code properties

- Only 1 loop can run in parallel at a time. Apart from a dedicated core responsible for executing code outside parallel loops, each core is either executing an iteration of the current loop or waiting for the start of the next one.
- Successive loop iterations are distributed to threads in a round-robin manner. Since each thread is pinned to a predefined core, and cores are organized in a unidirectional ring, successive iterations form a logical ring.
- Communication between cores executing a parallelized loop occurs only within sequential segments.
- Different sequential segments always access different shared data. HCCv3 only generates multiple sequential segments when there is no intersection of shared data. Consequently, instances of distinct sequential segments may run in parallel.
- At most 2 signals per sequential segment emitted by a given core can be in flight at any time. Hence, only 2 signals per segment need to be tracked by the ring cache.

This last property allows the elimination of unnecessary

`wait` instructions while keeping the architectural enhancement simple. Eliminating `wait`s allows a core to execute a later loop iteration than its successor (significantly boosting parallelism). Future iterations, however, produce signals that must be buffered. The last code property prevents a core from getting more than one “lap” ahead of its successor. So when buffering signals, each ring cache node only needs to recognize 2 types—those from the past and those from the future.

4.2. Code optimizations

In addition to conventional optimizations specifically tuned to extract Thread Level Parallelism (TLP) (e.g., code scheduling, method inlining, loop unrolling), HCCv3 includes ones that are essential for best performance of non-numerical programs on a ring-cache-enhanced architecture: aggressive splitting of sequential segments into smaller code blocks; identification and selection of small hot loops; and elimination of unnecessary `wait` instructions.

Sizing sequential segments poses a tradeoff. Additional segments created by splitting run in parallel with others, but extra segments entail extra synchronization, which adds communication overhead. Thanks to decoupling, HCCv3 can split aggressively to efficiently extract TLP. Note that segments cannot be split indefinitely—each shared location must be accessed by only 1 segment.

To identify small hot loops that are most likely to speed up when their iterations run in parallel, HCCv3 profiles the program being compiled using representative inputs. Instrumentation code emulates execution with the ring cache during profiling, which produces an estimate of time saved by parallelization. Finally, HCCv3 uses a loop nesting graph, annotated with the profiling results, to choose the most promising loops.

5. ARCHITECTURE ENHANCEMENTS

Adding a ring cache to a multicore architecture enables the proactive circulation of data and signals that boost parallelization. This section describes the design of the ring cache and its constituent ring nodes. The design is guided by the following objectives:

Low-latency communication. HELIX-RC relies on fast communication between cores in a multicore processor for synchronization and for data sharing between loop

iterations. Since low-latency communication is possible between physically adjacent cores in modern processors, the ring cache implements a simple unidirectional ring network.

Caching shared values. A compiler cannot easily guarantee whether and when shared data generated by a loop iteration will be consumed by other cores running subsequent iterations. Hence, the ring cache must cache shared data. Keeping shared data on local ring nodes provides quick access for the associated cores. As with data, it is also important to buffer signals in each ring node for immediate consumption.

Easy integration. The ring cache is a minimally-invasive extension to existing multicore systems, easy to adopt and integrate. It does not require modifications to the existing memory hierarchy or to cache coherence protocols.

With these objectives in mind, we now describe the internals of the ring cache and its interaction with the rest of the architecture.

5.1. Ring cache architecture

The ring cache architecture relies on properties of compiled code, which imply that the data involved in timing-critical dependences that potentially limit overall performance are both produced and consumed in the same order as loop iterations. Furthermore, a ring network topology captures this data flow, as sketched in Figure 4. The following paragraphs describe the structure and purpose of each ring cache component.

Ring node structure. The internal structure of a per-core ring node is shown in the right half of Figure 4. Parts of this structure resemble a simple network router. Unidirectional links connect a node to its two neighbors to form the ring backbone. Bidirectional connections to the core and private L1 cache allow injection of data into and extraction of data from the ring. There are 3 separate sets of data links and buffers. A primary set forwards data and signals between cores. Two other sets manage infrequent traffic for integration with the rest of the memory hierarchy (see Section 5.2). Separating these 3 traffic types simplifies the design and avoids deadlock. Finally, signals move in lockstep with forwarded data to ensure that a shared memory location is not accessed before the data arrives.

In addition to these router-like elements, a ring node also contains structures more common to caches. A set associative *cache array* stores all data values (and their tags) received by the ring node, whether from a predecessor node or from its associated core. The line size of this cache array is kept at one machine word. While the small line is contrary to typical cache designs, it ensures there will be no false data sharing by independent values from the same line.

The final structural component of the ring node is the *signal buffer*, which stores signals until they are consumed.

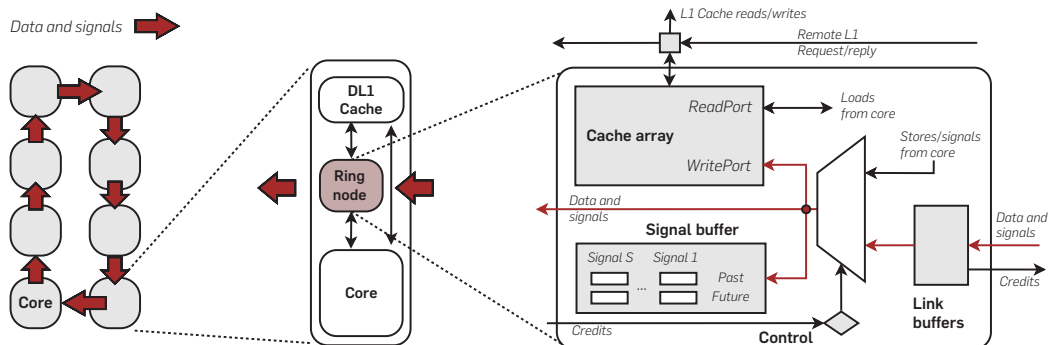
Node-to-node connection. The main purpose of the ring cache is to proactively provide many-to-many core communication in a scalable and low-latency manner. In the unidirectional ring formed by the ring nodes, data propagates by *value circulation*. Once a ring node receives an (address, value) pair, either from its predecessor, or from its associated core, it stores a local copy in its cache array and propagates the same pair to its successor node. The pair eventually propagates through the entire ring (stopping after a full cycle) so that any core can consume the data value from its local ring node, as needed.

This value circulation mechanism allows the ring cache to communicate between cores faster than reactive systems (like most coherent cache hierarchies). In a reactive system, data transfer begins once the receiver requests the shared data, which adds transfer latency to an already latency-critical code path. In contrast, a proactive scheme overlaps transfer latencies with computation to lower the receiver's perceived latency.

The ring cache prioritizes the common case, where data generated within sequential segments must propagate to all other nodes as quickly as possible. Assuming no contention over the network and single-cycle node-to-node latency, the design shown in Figure 4 allows us to bound the latency for a full trip around the ring to N clock cycles, where N is the number of cores. Each ring node prioritizes data received from the ring and stalls injection from its local core.

To eliminate delays to forward data between ring nodes, the number of write ports in each node's cache array must match the link bandwidth between two nodes. While this may seem like an onerous design constraint for the cache array, Section 6 shows that just one write port is sufficient to reap more than 99% of the ideal-case benefits.

Figure 4. Ring cache architecture overview. From left to right: overall system; single core slice; ring node internal structure.



To ensure correctness under network contention, the ring cache is sometimes forced to stall all messages (data and signals) traveling along the ring. The only events that can cause contention and stalls are ring cache misses and evictions, which may then need to fetch data from a remote L1 cache. While these ring stalls are necessary to guarantee correctness, they are infrequent.

The ring cache relies on credit-based flow control⁹ and is deadlock free. Each ring node has at least two buffers attached to the incoming links to guarantee forward progress. The network maintains the invariant that there is always at least one empty buffer per set of links somewhere in the ring. That is why a node only injects new data from its associated core into the ring when there is no data from a predecessor node to forward.

Node-core integration. Ring nodes are connected to their respective cores as the closest level in the cache hierarchy (Figure 4). The core's interface to the ring cache is through regular loads and stores for memory accesses in sequential segments.

As previously discussed, `wait` and `signal` instructions delineate code within a sequential segment. A thread that needs to enter a sequential segment first executes a `wait`, which only returns from the associated ring node when matching signals have been received from all other cores executing prior loop iterations. The signal buffer within the ring node enforces this. Specialized core logic detects the start of the sequential segment and routes memory operations to the ring cache. Finally, executing the corresponding `signal` marks the end of the sequential segment.

The `wait` and `signal` instructions require special treatment in out-of-order cores. Since they may have system-wide side effects, these instructions must issue non-speculatively from the core's store queue and regular loads and stores cannot be reordered around them. Our implementation reuses logic from load-store queues for memory disambiguation and holds a lightweight local fence in the load queue until the `wait` returns to the senior store queue. This is not a concern for in-order cores.

5.2. Memory hierarchy integration^a

The ring cache is a level within the cache hierarchy and as such must not break any consistency guarantees that the hierarchy normally provides. Consistency between the ring cache and the conventional memory hierarchy results from the following invariants: (i) shared memory can only be accessed within sequential segments through the ring cache (compiler-enforced); (ii) only a uniquely assigned *owner* node can read or write a particular shared memory location through the L1 cache on a ring cache miss (ring cache-enforced); and (iii) the cache coherence protocol preserves the order of stores to a memory location through a particular L1 cache.

Sequential consistency. To preserve the semantics of a parallelized single-threaded program, memory operations on shared values require sequential consistency. The ring

cache meets this requirement by leveraging the unidirectional data flow guaranteed by the compiler. Sequential consistency must be preserved when ring cache values reach lower-level caches, but the consistency model provided by conventional memory hierarchies is weaker. We resolve this difference by introducing a single serialization point per memory location, namely a unique *owner* node responsible for all interactions with the rest of the memory hierarchy. When a shared value is moved between the ring cache and L1 caches (owing to occasional ring cache load misses and evictions), only its owner node can perform the required L1 cache accesses. This solution preserves existing consistency models with minimal impact on performance.

Cache flush. Finally, to guarantee coherence between parallelized loops and serial code between loop invocations, each ring node flushes the dirty values of memory locations it owns to L1 once a parallel loop has finished execution. This is equivalent to executing a distributed fence at the end of loops. In a multiprogram scenario, signal buffers must also be flushed/restored at program context switches.

6. EVALUATION^b

By co-designing the compiler along with the architecture, HELIX-RC more than triples the performance of parallelized code when compared to a compiler-only solution. This section investigates HELIX-RC's performance benefits and their sensitivity to ring cache parameters. We confirm that the majority of speedups come from decoupling all types of communication and synchronization. We conclude by analyzing the remaining overheads of the execution model.

6.1. Experimental setup

We ran experiments on 2 sets of architectures. The first relies on a conventional memory hierarchy to share data among the cores. The second relies on the ring cache.

Simulated conventional hardware. We simulate a multi-core in-order x86 processor by adding multiple-core support to the XIOSim simulator. We also simulate out-of-order cores modeled after Intel Nehalem.

The simulated cache hierarchy has 2 levels: a per-core 32 KB, 8-way associative L1 cache and a shared 8 MB 16-bank L2 cache. We vary the core count from 1 to 16, but do not vary the amount of L2 cache with the number of cores, keeping it at 8 MB for all configurations. Also scaling cache size would make it difficult to distinguish the benefits of parallelizing a workload from the benefits of fitting its working set into the larger cache, causing misleading results. Finally, we use DRAMSim2 for cycle-accurate simulation of memory controllers and DRAM.

We extended XIOSim with a cache coherence protocol assuming an optimistic cache-to-cache latency of 10 clock cycles. This 10-cycle latency is optimistically low even compared to research prototypes of low-latency coherence.¹¹ We only use this low-latency model to simulate conventional hardware, and later (Section 6.2) shows that low latency alone is not enough to compensate for the lazy nature of its

^a This feature may add one multiplexer delay to the critical delay path from the core to L1.

^b Most cache coherence protocols (including Intel, AMD, and ARM implementations) provide this minimum guarantee.

coherence protocol.

Simulated ring cache. We extended XIOSim to simulate the ring cache as described in Section 5. We used the following configuration: a 1 KB 8-way associative array size, one-word data bandwidth, five-signal bandwidth, single-cycle adjacent core latency, and two cycles of core-to-ring-node injection latency to minimally impact the already delay-critical path from the core to the L1 cache. A sensitivity analysis of these parameters as well as the evaluation of the ring cache in out-of-order cores can be found in.⁴ We use a simple bit mask as the hash function to distribute memory addresses to their owner nodes. To avoid triggering the cache coherence protocol, all words of a cache line have the same owner. Lastly, XIOSim simulates changes made to the core to route memory accesses either to the attached ring node or to the private L1.

Benchmarks. We use 10 out of the 15 C benchmarks from the SPEC CPU2000 suite: 4 floating point (CFP2000) and 6 integer benchmarks (CINT2000). For engineering reasons, the data dependence analysis that HCCv3 relies on⁴ requires either too much memory or too much time to handle the rest. This limitation is orthogonal to the results described in this paper.

Compiler. We extended the Intermediate Language Distributed Just-In-Time (ILDJIT) compilation framework,³ version 1.1, to use LLVM 3.0 for backend machine code generation. We generated both single- and multi-threaded versions of the benchmarks. The single-threaded programs are the unmodified versions of benchmarks, optimized (O3) and generated by LLVM. This code outperforms GCC 4.8.1 by 8% on average and under-performs ICC 14.0.0 by 1.9%. The multi-threaded programs were generated by HCCv3 and the HELIX compiler (i.e., compiler-only solution) to run on ring-cache-enhanced and conventional architectures, respectively. Both compilers produce code automatically and do not require any human intervention. During compilation, they use SPEC training inputs to select the loops to parallelize.

Measuring performance. We compute speedups relative to sequential simulation. Both single- and multi-threaded runs use reference inputs. To make simulation feasible, we simulate multiple phases of 100 M instructions as identified by SimPoint.

6.2. Speedup analysis^c

In our 16-core processor evaluation system, HELIX-RC boosts the performance of sequentially-designed programs (CINT2000), assumed not to be amenable to parallelization. Figure 5 shows that HELIX-RC raises the geometric mean of speedups for these benchmarks from 2.2× for a compiler-only solution to 6.85×.

HELIX-RC not only maintains the performance of a compiler-only solution on numerical programs (SPEC CFP2000),

^c As an aside, automatic parallelization features of ICC led to a geometric slowdown of 2.6% across SPEC CINT2000 benchmarks, suggesting ICC cannot parallelize non-numerical programs. These speedups are possible even with a cache coherence latency of conventional processors (e.g., 75 cycles).

but also increases the geometric mean of speedups for CFP2000 benchmarks from 11.4× to almost 12×.

We now turn our attention to understanding where the speedups come from.

Communication. Speedups obtained by HELIX-RC come from decoupling both synchronization and data communication from computation in loop iterations, which significantly reduces communication overhead, allows the compiler to split sequential segments into smaller blocks, and cuts down the critical path of the generated parallel code. Figure 6 compares the speedups gained by multiple combinations of decoupling synchronization, register-, and memory-based communication. As expected, fast register transfers alone do not provide much speedup since most in-register dependences can be satisfied by re-computing the shared variables involved.⁴ Instead, most of the speedups come from decoupling communication for both synchronization and memory-carried actual dependences. To the best of our knowledge, HELIX-RC is the only solution that accelerates all 3 types of transfers for actual dependences.

Figure 5. HELIX-RC triples the speedup obtained by a compiler-only solution for SPEC INT benchmarks. Speedups are relative to sequential program execution.

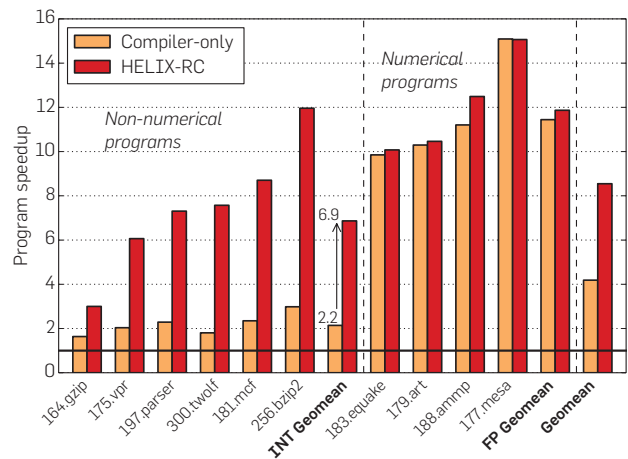
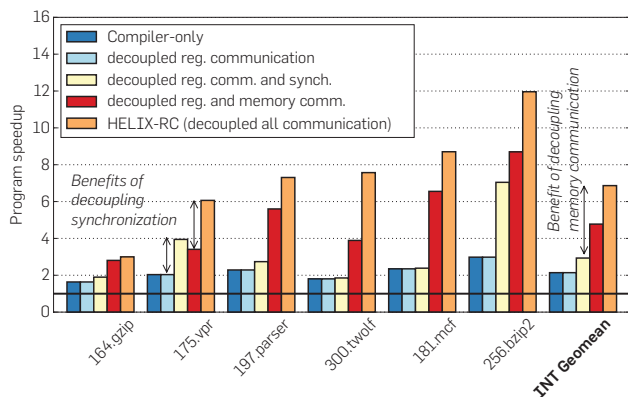


Figure 6. Breakdown of benefits of decoupling communication from computation.



Sequential segments. While more splitting offers higher TLP (more sequential segments can run in parallel), it also requires more synchronization at run time. Hence, the high synchronization cost for conventional multicores discourages aggressive splitting of sequential segments. In contrast, the ring cache enables aggressive splitting to maximize TLP.

To analyze the relationship between splitting and TLP, we computed the number of instructions that execute concurrently for the following 2 scenarios: (i) conservative splitting constrained by a contemporary multicore processor with high synchronization penalty (100 cycles) and (ii) aggressive splitting for HELIX-RC with low-latency communication (<10 cycles) provided by the ring cache. In order to compute TLP independent of both the communication overhead and core pipeline advantages, we used a simple abstracted model of a multicore system that has no communication cost and is able to execute 1 instruction at a time. Using the same set of loops chosen by HELIX-RC and used in Figure 5, TLP increased from 6.4 to 14.2 instructions with aggressive splitting. Moreover, the average number of instructions per sequential segment dropped from 8.5 to 3.2 instructions.

Coverage. Despite all the loop-level speedups possible via decoupling communication and aggressively splitting of sequential segments, Amdahl's law states that program coverage dictates the overall speedup of a program. Prior parallelization techniques have avoided selecting loops with small bodies because communication would slow down execution on conventional processors.^{5,20} Since HELIX-RC does not suffer from this problem, the compiler can freely select small hot loops to cover almost the entirety of the original program.

6.3. Analysis of overhead

To understand areas for improvement, we categorize every overhead cycle preventing ideal speedup. Figure 7 shows the results of this categorization for HELIX-RC, again implemented on a 16-core processor.

Most importantly, the small fraction of *communication* overheads suggests that HELIX-RC successfully eliminates the core-to-core latency for data transfer in most benchmarks. For several benchmarks, notably 175.vpr, 300.twolf, 256.bzip2, and 179.art, the major source of overhead is the low number of iterations per parallelized loop (*low trip count*). While many hot loops are frequently invoked, low

iteration count (ranging from 8 to 20) leads to idle cores. Other benchmarks such as 164.gzip, 197.parser, 181.mcf, and 188.ammf suffer from dependence waiting due to large sequential segments. Finally, HCCv3 must sometimes add a large number of `wait` and `signal` instructions (i.e., many sequential segments) to increase TLP, as seen for 164.gzip, 197.parser, 181.mcf, and 256.bzip2.

7. RELATED WORK

To compare HELIX-RC to a broad set of related work, Table 1 summarizes different parallelization schemes proposed for non-numerical programs organized with respect to the types of communication decoupling implemented (register vs. memory) and the types of dependences targeted (actual vs. false). HELIX-RC covers the entire design space and is the only one to decouple memory accesses from computation for actual dependences.

7.1. Multiscalar register file

Multiscalar processors¹⁹ extract both Instruction Level Parallelism (ILP) and TLP from an ordinary application. While a ring cache's structure resembles a Multiscalar register file, there are fundamental differences. For the Multiscalar register file, there is a fixed and relatively small number of shared elements that must be known at compile time. Furthermore, the Multiscalar register file cannot handle memory updates by simply mapping memory to a fixed number registers without a replacement mechanism. In contrast, the ring cache does not require compile-time knowledge to handle an arbitrary number of elements shared between cores (i.e., memory locations allocated at runtime) and can readily handle register updates by deallocating a register to a memory location. In other words, HELIX-RC proposes to use a distributed cache to handle both register and memory updates.

7.2. Cache coherence protocols

The ring cache addresses an entirely different set of communication demands. Cache coherence protocols target relatively small amounts of data shared infrequently between cores. Hence, cores can communicate lazily, but the resulting communication almost always lies in the critical sequential chain. In contrast, the ring cache targets frequent and time-critical data sharing between cores.

7.3. On-chip networks

Figure 7. Breakdown of overheads that prevent achieving ideal speedup.

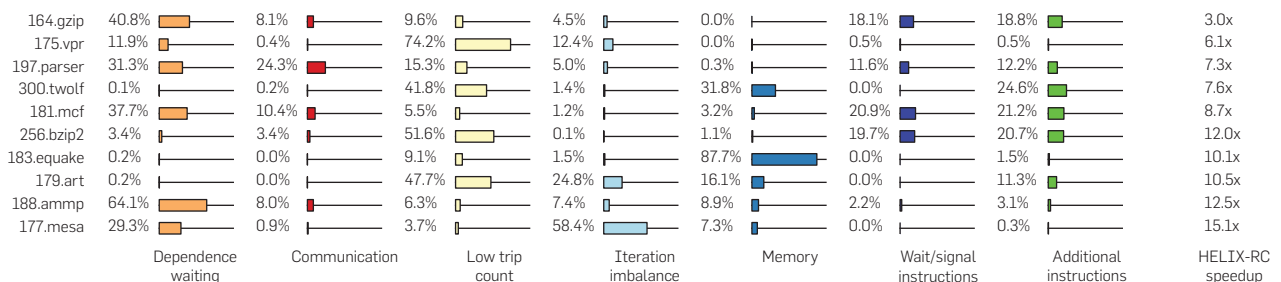


Table 1. Only HELIX-RC decouples communication for all types of dependences.

| | Actual dependences | False dependences |
|----------|--|--|
| Register | HELIX-RC , Multiscalar, TRIPS, T3 | HELIX-RC , Multiscalar, TRIPS, T3 |
| Memory | HELIX-RC | HELIX-RC , TLS-based approaches, Multiscalar, TRIPS, T3 |

While On-Chip-Networks (OCNs) can take several forms, they commonly implement reactive coherence protocols^{18,21,24,25} that do not fulfill the low-latency communication requirements of HELIX-RC. Scalar operand networks²² somewhat resemble a ring cache to enable tight coupling between known producers and consumers of specific operands, but they suffer from the same limitations as the Multiscalar register file. Hence, HELIX-RC implements a relatively simple OCN, but supported by compiler guarantees and additional logic to implement automatic forwarding.

7.4. Off-chip networks

Networks that improve bandwidth between processors have been studied extensively.¹⁷ While they work well for Cyclic Multithreading (CMT) parallelization techniques that require less frequent data sharing, there is less overall parallelism. Moreover, networks that target chip-to-chip communication do not meet the very different low-latency core-to-core communication demands of HELIX-RC.⁹ Our results show HELIX-RC is much more sensitive to latency than to bandwidth.

7.5. Non-commodity processors

Multiscalar,¹⁹ TRIPS,¹⁶ and T3 (Ref. Robotmil *et al.*¹⁵) are polymorphous architectures that target parallelism at different granularities. They differ from HELIX-RC in that (i) they require a significantly larger design effort and (ii) they only decouple register-to-register communication and/or false memory dependence communication by speculating.

An iWarp system² implements special-purpose arrays that execute fine- and coarse-grained parallel numerical programs. However, without an efficient broadcast mechanism, iWarp's fast communication cannot reach the speedups offered by HELIX-RC.

7.6. Automatic parallelization of non-numerical programs

Several automatic methods to extract TLP have demonstrated respectable speedups on commodity multicore processors for non-numerical programs.^{5,8,13,23} All of these methods transform loops into parallel threads. Decoupled Software Pipelining (DSWP)¹³ reduces sensitivity to communication latency by restructuring a loop to create a pipeline among the extracted threads with unidirectional communication between pipeline stages. Demonstrated both on simulators and on real systems, DSWP performance is largely insensitive to latency. However, significant restructuring of the loop makes speedups difficult to predict and generated

code can sometimes be slower than the original. Moreover, DSWP faces the challenges of selecting appropriate loops to parallelize and keeping the pipeline balanced at runtime. While DSWP-based approaches focus more on restructuring loops to hide communication latency,^{8,13} HELIX-RC proposes an architecture-compiler co-design strategy that selects the most appropriate loops for parallelization.

Combining DSWP with HELIX-RC has the potential to yield significantly better performance than either alone. DSWP cannot easily scale beyond 4 cores¹⁴ without being combined with approaches that exploit parallelism among loop iterations (e.g., DOALL).⁸ While DSWP + DOALL can scale beyond several cores, DOALL parallelism is not easy to find in non-numerical code. Instead, DSWP + HELIX-RC presents an opportunity to parallelize a much broader set of loops.

Several TLS-based techniques,^{7,10,20} including Stanford Hydra, POSH, and STAMPede, combine hardware-assisted Thread Level Speculation (TLS) with compiler optimizations to manage dependences between loop iterations executing in different threads. When the compiler identifies sources and destinations of frequent dependences, it synchronizes using `wait` and `signal` primitives; otherwise, it uses speculation. HELIX-RC, on the other hand, optimizes code assuming all dependences are actual. While we believe adding speculation may help HELIX-RC, Figure 5 shows decoupled communication already yields significant speedups without misspeculation overheads.

8. CONCLUSION

HELIX-RC shows how to *accelerate non-numerical programs* by exploiting parallelism between the iterations of their small loops. Successfully mapping the iterations of such loops onto multiple cores of a single chip requires a *low-latency*, broadcast interconnect between cores. This interconnect needs to be *proactive* (so that communication starts as soon as data is generated), and it must be able to update memory locations stored in each core's private cache.

8.1. Accelerating non-numerical programs to catch up with hardware evolution

Adding multiple cores to a single chip has been proposed, studied, and realized in products since the 90's (Ref. Olukotun *et al.*¹²), but the majority of these cores are still under-utilized even after more than 15 years' effort in both compiler and programming language research. Having reached the "ILP wall," industry now relies on these multiple cores to gain performance from each system. However, successful uses of multiple cores exist only when the goal is maximizing throughput combined with massive data parallelism or parallelism among multiple programs, as is available in Graphics Processing Unit (GPU) computing or within data centers. On the other hand, if single program performance is the target and there is little or no data parallelism available (e.g., non-numerical programs running on mobile phones or client computers), then only a few cores are actually used, leaving the majority of them under-utilized.¹ Our work shows how to actually take advantage of the cores that are available within a single chip when running non-numerical programs, highlighting the great potential of including


hardware support for a proactive, cache-based, low-latency core-to-core interconnect.

8.2. Transforming parallelism into performance requires low-latency communication

Our work demonstrates the fundamental value of having a low-latency interconnect to boost the performance of complex, non-numerical programs. The dependence between communication latency and performance of a program has already been observed in high-performance computing domains.¹⁷ Moreover, prior work on on-chip networks has shown the value of a low-latency interconnect both for programs with regular control and data flows^{22,25} and for a novel research architecture.^{6,21} Our work is the first to demonstrate the value of a cache-based, low latency interconnect between cores of commodity processors for accelerating complex, non-numerical programs running on a chip.

8.3. From reactive hardware-driven to proactive software-driven cache communication

HELIX-RC has the potential to influence the adoption of proactive, cache-based, and one-to-many interconnects in commodity processors. To quantify the need for such solutions, we measured the communication latency between adjacent cores in several generations of Intel commodity processors. As highlighted in Figure 1a, conventional reactive solutions have latencies of around 100 cycles. The figure shows that, among the five generations of Intel processors we considered, adjacent core latency bounces around 100 cycles without a monotonic trend over time. This suggests that there is no reason to expect conventional solutions (reactive hardware-driven) to improve in the future.

HELIX-RC motivates shifting inter-core communication mechanisms towards alternative cache-based solutions, in which a compiler identifies for the hardware the code that will generate shared data. The architecture, for its part, will proactively communicate modified values to make them locally accessible by other cores. This allows a drastic cut in the latency of remote data access, which, therefore, allows a parallelizing compiler to take advantage of the substantial latent parallelism between the iterations of small loops. 

References

1. Blake, G., Dreslinski, R.G., Mudge, T., Flautner, K. Evolution of thread-level parallelism in desktop applications. In *ISCA* (2010).
2. Borkar, S., Cohn, R., Cox, G., Gleason, S., Gross, T., Kung, H.T., Lam, M., Moore, B., Peterson, C., Pieper, J., Rankin, L., Tseng, P.S., Sutton, J., Urbanski, J., Webb, J. iWarp: An integrated solution to high-speed parallel computing. *Supercomputing* (1988).
3. Campanoni, S., Agosta, G., Reghizzi, S.C., Biagio, A.D. A highly flexible, parallel virtual machine: Design and experience of ILDJIT. *Software: Practice and Experience* (2010).
4. Campanoni, S., Brownell, K., Kanev, S., Jones, T.M., Wei, G.-Y., Brooks, D. HELIX-RC: An architecture-compiler co-design for automatic parallelization of irregular programs. In *ISCA* (2014).
5. Campanoni, S., Jones, T., Holloway, G., Reddi, V.J., Wei, G.-Y., Brooks, D. HELIX: Automatic Parallelization of Irregular Programs for Chip Multiprocessing. In *CGO* (2012).
6. Ceze, L., Tuck, J., Torrellas, J., Cascaval, C. Bulk disambiguation of speculative threads in multiprocessors. In *ISCA* (2006).
7. Hammond, L., Hubbert, B.A., Siu, M., Prabhu, M.K., Chen, M.K., Olukotun, K. The Stanford Hydra CMP. *IEEE Micro* (2000).
8. Huang, J., Raman, A., Jablin, T.B., Zhang, Y., Hung, T.-H., August, D.I. Decoupled software pipelining creates parallelization opportunities. In *CGO* (2010).
9. Jerger N.E., Peh, L.-S. *On-Chip Networks*. Synthesis Lectures on Computer Architecture. Morgan & Claypool, 2009.
10. Liu, W., Tuck, J., Ceze, L., Ahn, W., Strauss, K., Renau, J., Torrellas, J. POSH: A TLS compiler that exploits program structure. In *PPoPP* (2006).

11. Martin, M.M.K. *Token Coherence*. PhD thesis, University of Wisconsin-Madison, 2003.
12. Olukotun, K., Nayfeh, B.A., Hammond, L., Wilson, K., Chang, K. The case for a single-chip multiprocessor. *ASPLOS* (1996).
13. Ottoni, G., Rangan, R., Stoler, A., August, D.I. Automatic thread extraction with decoupled software pipelining. In *MICRO* (2005).
14. Rangan, R., Vachharajani, N., Ottoni, G., August, D.I. Performance scalability of decoupled software pipelining. In *ACM TACO* (2008).
15. Robatmil, B., Li, D., Esmaeilzadeh, H., Govindan, S., Smith, A., Putnam, A., Burger, D., Keckler, S.W. How to Implement Effective Prediction and Forwarding for Fusible Dynamic Multicore Architectures. In *HPCA* (2013).
16. Sankaralingam, K., Nagarajan, R., Liu, H., Kim, C., Huh, J., Ranganathan, N., Burger, D., Keckler, S.W., McDonald, R.G., Moore, C.R. TRIPS: A polymorphic architecture for exploiting ILP, TLP, and DLP. In *ACM TACO* (2004).
17. Scott, S.L. Synchronization and Communication in the T3E Multiprocessor. In *ASPLOS* (1996).
18. Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerma, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., Hanrahan, P. Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics* (2008).
19. Sohi, G.S., Breach, S.E., Vijaykumar, T.N. Multiscalar processors. In *ISCA* (1995).
20. Steffan, J.G., Colohan, C., Zhai, A., Mowry, T.C. The STAMPede approach to thread-level speculation. *ACM Transactions on Computer Systems* (2005).
21. Taylor, M.B., Kim, J., Miller, J., Wentzloff, D., Ghodrati, F., Greenwald, B., Hoffman, H., Johnson, P., Lee, J.-W., Lee, W., Ma, A., Saraf, A., Seneski, M., Shnidman, N., Strumpfen, V., Frank, M., Amarasinghe, S., Agarwal, A. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro* (2002).
22. Taylor, M.B., Lee, W., Amarasinghe, S.P., Agarwal, A. Scalar operand networks. *IEEE Transactions on Parallel Distributed Systems* (2005).
23. Tournavitis, G., Wang, Z., Franke, B., O'Boyle, M.F.P. Towards a holistic approach to auto-parallelization. In *PLDI* (2009).
24. van der Wijngaert, R.F., Mattson, T.G., Haas, W. Light-weight communications on Intel's single-chip cloud computer processor. *SIGOPS Operating Systems Review* (2011).
25. Wentzloff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C.-C., Brown, J.F. III, Agarwal, A. On-chip interconnection architecture of the tile processor. *IEEE Micro* (2007).

Simone Campanoni, Northwestern University, Evanston, IL.
Kevin Brownell, Svilen Kanev, Gu-Yeon Wei, and David Brooks, Harvard University, Cambridge, MA.
Timothy M. Jones, University of Cambridge, England.

CAREERS

Auburn University **Department of Computer Science** **and Software Engineering (CSSE)** **Multiple Faculty Positions in Cybersecurity**

CSSE invites applications for multiple tenure-track faculty positions to begin in Fall 2018 or later. Candidates at the Assistant Professor level will be given preference, however outstanding candidates at senior levels will also be considered. A Ph.D. degree in computer science, software engineering or a closely related field must be completed by the start of appointment. Excellent communication skills are required. We are interested in candidates specializing in all areas related to security, such as *AI/machine learning applications to security, blockchain, cryptocurrency, cyberidentity, cyberinfrastructure and critical infrastructure protection, digital forensics, reverse engineering, secure cloud, mobile systems, networks and operating systems, secure software engineering, and securing the Internet of Things.*

CSSE is home to the Auburn Cyber Research Center (<http://cyber.auburn.edu>), and is affiliated with the McCrary Institute for Critical Infrastructure Protection and Cyber Systems (<http://mccrary.auburn.edu>). The department has 21 full-time tenure-track faculty members and supports strong M.S. and Ph.D. programs (with a new M.S. in Cybersecurity Engineering projected to start in Fall 2018). Faculty research areas include artificial intelligence, architecture, computational biology, computer science education, data science, energy-efficient systems, human-computer interaction, Internet of Things, learning science, machine learning, modeling and simulation, multi-agent systems, networks, security, software engineering and wireless engineering.

Auburn University is one of the nation's premier public land-grant institutions. It is ranked 46th among public universities in the U.S. News and World Report 2018 Rankings. It is nationally recognized for its commitment to academic excellence, a positive work environment, student engagement, and its beautiful campus. Auburn residents enjoy a thriving community, recognized as one of the "best small towns in America." The city is located on the rapidly developing I-85 corridor between Atlanta, GA, and Montgomery, AL. The Auburn City school system is ranked as one of the top school systems in the nation and the state. A nationally recognized hospital, East Alabama Medical Center, is located close by in Opelika. The Auburn-Opelika metropolitan area has a population of over 150,000.

Applicants should submit a cover letter, curriculum vita, research vision, teaching philosophy, and the names of references through the faculty hiring link on the department home page <http://www.eng.auburn.edu/csse>. There is no application deadline. Application review will begin in November. Selected candidates must be able to meet eligibility requirements to work legally in the United States at the time of appointment for

the proposed term of employment. Auburn University is an EEO/Vet/Disability Employer.

Barnard College **Senior Endowed Chair and Director of** **the Center for Computational Science**

Barnard College, a premier liberal arts college in the City of New York and the nation's most selective women's college, seeks a scholar and leader to serve as Inaugural Endowed Chair in Computer Science and Director of the Center for Computational Science. This individual will build a department of computer science at the College that works in close collaboration with the Department of Computer Science in Columbia University's Fu Foundation School of Engineering and Applied Science. In this newly-created role, the Chair and Director will have the opportunity to launch a new academic program in Computer Science and frame programming and content for a new Center for Computational Science. The endowed chair position will be tenured in the Barnard Department of Mathematics until a department of computer science is formally established. The Chair and Director is expected to be in place no later than September 2018.

Barnard benefits from close collaborations with academic departments across Columbia University, and it is expected that this individual will have an affiliate position and intellectual home in both the Computer Science Department and at the Data Sciences Institute at Columbia University. In addition to curricular development, the endowed chair position will have the resources to recruit junior faculty and post-doctoral scholars and stimulate interactions with units across Barnard, many of which have quantitatively and computationally oriented components, and with Columbia.

Barnard seeks a leader in the field of computer science with a distinguished record of scholarship and is open to candidates from a variety of research areas. The ability to develop curriculum, an understanding of the importance of collaboration across disciplines, and a deep commitment to the undergraduate experience, the importance of women's colleges, and excellent communication skills are required.

Applications should be submitted electronically and include the following: curriculum-vitae including a publication list, a description of research accomplishments, a statement of research and teaching interests and plans, contact information for three experts who can provide letters of recommendation, and up to three pre/reprints of scholarly work.

Inquiries, nominations, and applications should be sent in strict confidence to:

Jane McMahon, Managing Associate
Isaacson, Miller
www.imsearch.com/6376

Barnard College is an Equal Opportunity Employer. Barnard does not discriminate due to race, color, creed, religion, sex, sexual orientation, gender and/or gender identity or expression, marital or parental status, national origin, ethnicity, citizenship status, veteran or military status, age, disability, or any other legally protected basis, and to the extent permitted by law. Qualified candidates of diverse ethnic and racial backgrounds are encouraged to apply for vacant positions at all levels.

Boston College **Assistant Professor of the Practice or Lecturer**

The Computer Science Department of Boston College aims to grow substantially over the next several years, and will seek to fill faculty positions at all levels. We invite applications for a full-time, non-tenure-track faculty position, beginning in the 2018-2019 academic year. Candidates should be committed to excellence in undergraduate education, and should be able to both teach a broad variety of undergraduate computer science courses, and to 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 include a Ph.D. in Computer Science or closely related discipline. Candidates who have only attained a Master's degree would be eligible for the title of Lecturer.

Application review begins October 1, 2017. See www.cs.bc.edu for more information.

To apply go to <http://apply.interfolio.com/44984>.

Boston College **Associate or Full Professor** **of Computer Science**

The Computer Science Department of Boston College aims to grow substantially over the next several years, and will seek to fill faculty positions at all levels. We invite applications for a senior-level (Associate or Full Professor) position, starting in the 2018-2019 academic year. The successful candidate is expected to play a leadership role in the creation of a Data Science program in conjunction with the new interdisciplinary Institute for Integrated Science and Society, recently announced at Boston College, and will also participate in shaping the future of our growing department.

Applicants must have a Ph.D. in Computer Science or closely related discipline, a record of strong research accomplishment and external funding, and a commitment to quality in undergraduate and graduate education. Preference will be given to candidates whose research is in the areas of high-performance data mining / machine

learning or data visualization, particularly those with a history of interdisciplinary collaboration, but outstanding candidates in all research areas will be considered.

Application review begins October 1, 2017. See www.cs.bc.edu for more information.

To apply go to <http://apply.interfolio.com/44982>

Boston College Tenure Track Assistant Professor in Computer Science

The Computer Science Department of Boston College aims to grow substantially over the next several years and will seek to fill faculty positions at all levels. We invite applications for one or more tenure-track faculty positions at the rank of Assistant Professor, beginning in the 2018-2019 academic year. Successful candidates will be expected to develop strong research programs that can attract external research funding. The search will focus on candidates who can participate in cross-disciplinary research in conjunction with the new Institute for Integrated Science and Society recently announced at Boston College, in the areas of high-performance data mining / machine learning, systems / networks, data visualization, and human-computer interaction. However, outstanding candidates in all research areas will be considered.

Minimum requirements for all positions include a Ph.D. in Computer Science or closely related discipline, an energetic research program that promises to attract external funding, and

a commitment to quality in undergraduate and graduate education.

Application review begins October 1, 2017. See www.cs.bc.edu for more information.

To apply go to <http://apply.interfolio.com/44980>

Bowling Green State University Assistant Professor - Computer Science - Cyber Security & Digital Forensics

Department of Computer Science: Assistant Professor, Bowling Green State University. Tenure-track faculty position available August 2018.

Responsibilities: Candidates must have potential for excellence in research and a strong commitment to teaching at both the undergraduate and graduate levels.

Minimum Qualifications: Area of specialization within Cyber Security and Digital Forensics are open. A Ph.D. in Computer Science or related field is required; advanced ABD candidates will be considered but must complete requirements for the Ph.D. prior to employment.

For a complete job description & instructions on how to apply for this position visit <https://bgsu.hiretouch.com/> or contact the Office of Human Resources, BGSU. Application deadline is January 5, 2018.

Background check is required for employment.

BGSU is an AA/EEO/Vet employer. We encourage applications from women, minorities, veterans, and persons with disabilities regardless of age, gender identity, genetic information, religion, or sexual orientation.

Carnegie Mellon University Faculty Hiring

The School of Computer Science consists of seven departments, spanning a wide range of topics in computer science and the application of computers to real-world systems. Faculty positions are specific to each department, though in certain cases, joint positions are also possible.

We are seeking tenure, research, and systems track faculty candidates with a strong interest in research, an earned Ph.D., and outstanding academic credentials. Candidates for tenure track appointments should also have a strong interest in graduate and undergraduate education.

We are also seeking teaching track faculty candidates. You should have a Ph.D. in Computer Science or a related computing discipline, a background of demonstrated excellence and dedication to teaching, the ability to collaborate with other faculty in a fast-paced environment, and must be prepared to teach in a wide variety of settings, including large undergraduate lecture courses and classes delivered in non-traditional formats.

Candidates with a commitment toward building an equitable and diverse scholarly community are particularly encouraged to apply. We are very interested in applications from candidates who have a demonstrated track record in mentoring and nurturing women and students from groups traditionally underrepresented in computer science.

To ensure full consideration of your application, please submit all materials no later than December 15, 2017. In your cover letter, please indicate clearly the department(s) you are applying



MULTIPLE FACULTY POSITIONS Department of Electrical and Systems Engineering

The School of Engineering and Applied Science at the University of Pennsylvania is growing its faculty by 33% over the next five years. As part of this initiative, the **Department of Electrical and Systems Engineering** is engaged in an aggressive, multi-year hiring effort for multiple tenure-track positions at all levels. Candidates must hold a Ph.D. in Electrical Engineering, Computer Engineering, Systems Engineering, or related area. The department seeks individuals with exceptional promise for, or proven record of, research achievement, who will take a position of international leadership in defining their field of study, and excel in undergraduate and graduate education. Leadership in cross-disciplinary and multi-disciplinary collaborations is of particular interest. We are interested in candidates in all areas that enhance our research strengths in

1. **Nanodevices and nanosystems** (nanoelectronics, MEMS/NEMS, powerelectronics, nanophotonics, integrated devices and systems at nanoscale),
2. **Circuits and computer engineering** (analog, RF, mm-wave, digital circuits, emerging circuit design, computer engineering, IoT, embedded and cyber-physical systems), and
3. **Information and decision systems** (control, optimization, robotics, data science, network science, communications, information theory, signal processing, markets and social systems).

Prospective candidates in all areas are strongly encouraged to address large-scale societal problems in energy, transportation, health, food and water, economic and financial networks, critical infrastructure, and national security. We are especially interested in candidates whose interests are aligned with the school's strategic plan, <http://www.seas.upenn.edu/PennEngineering2020/>

Diversity candidates are strongly encouraged to apply. Interested persons should submit an online application at <http://www.ese.upenn.edu/faculty-positions> and include curriculum vitae, statement of research and teaching interests, and at least three references. Review of applications will begin on December 1, 2017.

*The University of Pennsylvania is an Equal Opportunity Employer.
Minorities/Women/Individuals with Disabilities/Veterans are encouraged to apply.*

ROCHESTER INSTITUTE OF TECHNOLOGY Rochester, New York

Computing and Information Sciences - Multiple Openings for Fall 2018

The B. Thomas Golisano College of Computing and Information Sciences at the Rochester Institute of Technology invites applications and nominations for the following faculty positions:

- Tenure-track assistant professors (2) in Computer Science: all areas (#3444BR)
- Tenure-track assistant professor in Computing Science: cybersecurity including cryptography (#3442BR)
- Tenure-track assistant professor in Computing Security (#3213BR)
- Tenure-track assistant professor in Graduate Studies and Research: machine learning, data analytics, and their applications (BR#3463)
- Tenure-track assistant professor Interactive Games and Media: game artificial intelligence, real-time game graphics programming, game architecture and systems, virtual reality/augmented reality in games (#3414BR)
- Lecturer in Computer Science to teach in the areas of introductory and core Computer Science (#3445BR)
- Lecturer in Computing Security to teach introductory computer science, fundamentals of computing security, cryptography, systems and network security (#3418BR)
- Lecturer and Visiting Lecturers in Interactive Games and Media: game artificial intelligence, real-time game graphics programming, introductory game programming, data structures and algorithms, game development environments such as Unity or Unreal (#3413BR)
- Lecturer in Software Engineering (BR#3374)

Candidates should visit <http://careers.rit.edu/faculty> and refer to the BR numbers listed above for specific information about the positions and the application process. Refer to www.rit.edu for information about RIT and the B. Thomas Golisano College of Computing and Information Sciences.

The B. Thomas Golisano College of Computing and Information Sciences is the largest of RIT's nine colleges and has an enrollment of over 3100 undergraduate students and 850 graduate students. The college, with over 140 faculty members, is housed in a modern facility equipped with numerous teaching and research laboratories. The college is home to the departments of Computer Science, Computing Security, Information Sciences and Technologies, Software Engineering, the School of Interactive Games and Media, and a college-wide PhD program, providing many opportunities for cooperation and research collaboration within and beyond the college.

RIT has been honored by *The Chronicle of Higher Education* as one of the "Great Colleges to Work For" for four years. RIT is a National Science Foundation ADVANCE Institutional Transformation site. RIT is responsive to the needs of dual-career couples by our membership in the Upstate NY HERC. RIT does not discriminate. RIT is an equal opportunity employer that promotes and values diversity, pluralism, and inclusion. For more information or inquiries, please visit RIT/TitleIX or the U.S. Department of Education at ED.Gov





FACULTY POSITIONS

Department of Computer Science

The Department of Computer Science at Virginia Tech (www.cs.vt.edu) seeks applicants for five faculty positions, including two tenure-track Assistant Professor positions in data analytics, a tenure-track assistant professor position in human-centered computing, and two open rank positions in cybersecurity. Candidates must have a Ph.D. in computer science or related field at the time of appointment and a rank-appropriate record of scholarship and collaboration in computing research, broadly defined. Successful candidates should give evidence of commitment to issues of diversity in the campus community. Tenured and tenure-track faculty will be expected to teach graduate and undergraduate courses, mentor graduate students, and develop a high quality research program.

ASSISTANT PROFESSOR IN DATA ANALYTICS - Blacksburg, VA. Candidates with research depth and breadth in data analytics, data mining, machine learning, deep learning, artificial intelligence, text mining, natural language processing, information retrieval, interactive visual analytics, data visualization, high-performance analysis, social informatics, or data science are encouraged to apply. Candidates working at the intersection of data analytics and other computing or application domains - such as cyber-security, urban computing, health analytics, bioinformatics, and distributed and IoT systems - are also encouraged to apply. Successful candidates should be able to demonstrate an interest in initiating and sustaining collaborations within computing as well as with data domain scientists. Successful candidates will have the opportunity to engage in transdisciplinary research, curriculum, and outreach initiatives with other university faculty working in the Data & Decisions destination area, one of several new university-wide initiatives at Virginia Tech (provost.vt.edu/destination-areas). Data & Decisions is focused on advancing the human condition and society with better decisions through data. Faculty collaborating in this area integrate data analytics and decision sciences across transdisciplinary research and curriculum efforts at Virginia Tech. Candidates with demonstrated experience in interdisciplinary teaching or research that aligns with the Data and Decisions vision (provost.vt.edu/destination-areas/da-overview/da-data.html) are especially encouraged to apply. Successful candidates will also have opportunities to collaborate with numerous research centers on campus, including the Discovery Analytics Center (dac.cs.vt.edu), which leads big-data analytics research at Virginia Tech. Applications must be submitted online to jobs.vt.edu for posting #TR0170153. Applicant screening will begin on December 1, 2017. Inquiries should be directed to Dr. Chris North, Search Committee Chair, north@cs.vt.edu.

ASSISTANT PROFESSOR IN HUMAN-CENTERED COMPUTING - Blacksburg, VA. Candidates from any area related to human-computer interaction, user experience, or interactive computing are encouraged to apply. We especially encourage applicants with interests in novel interactive experiences and technologies—including immersive environments (virtual reality and augmented reality), multi-sensory displays, multi-modal input, visualization, visual analytics, human-robot interaction, game design, and creative technologies. The successful candidate will have the opportunity to engage in transdisciplinary research, curriculum, and outreach initiatives with other university faculty working in the Creativity & Innovation (C&I) Strategic Growth Area, one of several new university-wide initiatives at Virginia Tech (see provost.vt.edu/destination-areas). The C&I Strategic Growth Area is focused on empowering partners and stakeholders to collaborate on creativity, innovation, and entrepreneurship efforts that transcend disciplinary boundaries. Faculty working together in this area comprise a vibrant ecosystem that melds the exploration of innovative technologies and the design of creative experiences with best practices for developing impact-driven and meaningful outcomes and solutions. Candidates with demonstrated experience in interdisciplinary teaching or research that aligns with the C&I vision (provost.vt.edu/destination-areas/sga-overview/sga-creativity.html) are especially encouraged to apply. The successful candidate will also have opportunities for collaboration in the interdisciplinary Center for Human-Computer Interaction (www.hci.vt.edu); the Institute for Creativity, Arts, and Technology (icat.vt.edu); and the Discovery Analytics Center (dac.cs.vt.edu). Applications must be submitted online to jobs.vt.edu for posting #TR0170152. Applicant screening will begin on December 1, 2017. Inquiries should be directed to Dr. Doug Bowman, Search Committee Chair, dbowman@vt.edu.

FACULTY POSITIONS IN CYBERSECURITY - Blacksburg, VA and National Capital Region (NCR). Candidates are sought for two positions, at any rank, with expertise in a broad range of cybersecurity topics, including but not limited to systems and software security, security analytics, human-centric security, formal methods, trustworthy computing, network security, cloud security, data security, and security issues in cyber-physical, IoT, or autonomous systems. Applicants interested in either the main campus in Blacksburg, VA, or in Virginia Tech's National Capital Region campus, with facilities in Falls Church and Arlington, VA, are encouraged to apply. These positions are part of a broad university initiative in integrated security (provost.vt.edu/destination-areas/da-overview/da-security.html). Virginia Tech's Integrated Security Destination Area (ISDA) is focused on understanding and fostering a world in which individuals, institutions, and nations are secured by technology and social systems that follow ethical principles and promote values of social justice. Faculty working together in this area are bringing a transdisciplinary approach to the complex range of human and systems security challenges. Successful candidates will have the opportunity to engage in transdisciplinary research, curriculum, and outreach with other faculty working in the ISDA. Candidates with demonstrated experience in interdisciplinary teaching or research that aligns with the ISDA vision are particularly encouraged to apply. In addition to ISDA collaborations, successful candidates will have the opportunity to work with a large group of cybersecurity faculty in CS and ECE (www.cyber.vt.edu). Applications must be submitted online to jobs.vt.edu for posting #TR0170145. Candidates interested in the National Capital Region (Northern Virginia) campus should clearly indicate that in an application cover letter.

The Department of Computer Science has 47 teaching faculty, including 42 tenured or tenure-track faculty, over 840 undergraduate majors, and more than 250 graduate students. Department annual research expenditures over the last four years average \$13 million. The department is in the College of Engineering, whose undergraduate program ranks 14th and graduate program ranks 27th among all U.S. engineering schools (*USN&WR*, 2017). Most of these positions are located at the main campus of Virginia Tech in Blacksburg, VA, in a region consistently ranked among the country's best places to live. One of the cybersecurity faculty positions may be at the university's Northern Virginia campus, with facilities in Falls Church and Arlington, VA.

These positions require occasional travel to professional meetings. Selected candidates must pass a criminal background check prior to employment.

Virginia Tech is an AA/EEO employer, committed to building a culturally diverse faculty; we strongly encourage applications from women and minorities.

to. You can learn more about our hiring plans and application instructions by visiting <http://www.cs.cmu.edu/employment-scs>.

For more information about the hiring priorities in a particular department, please visit a department site below: Computational Biology Department: <http://www.cbd.cmu.edu/tenure-track-faculty-positions/>

Computer Science Department: <https://www.csd.cmu.edu/careers/faculty-hiring>

Human-Computer Interaction Institute: <https://hcii.cmu.edu/careers/list>

Institute for Software Research: <http://www.isri.cmu.edu/jobs/index.html>

Language Technologies Institute: <http://lti.cs.cmu.edu/news/lti-hiring>

Machine Learning Department: http://www.ml.cmu.edu/Faculty_Hiring.html

Robotics Institute: <http://ri.cmu.edu/about/hiring-faculty-positions/>

Please send email to faculty-search@cs.cmu.edu with any questions.

Carnegie Mellon University shall abide by the requirements of 41 CFR §§ 60-1.4(a), 60-300.5(a) and 60-741.5(a). These regulations prohibit discrimination against qualified individuals based on their status as protected veterans or individuals with disabilities, and prohibit discrimination against all individuals based on their race, color, religion, sex, or national origin. Moreover, these regulations require that covered prime contractors and subcontractors take affirmative action to employ and advance in employment individuals without regard to race, color, religion, sex, national origin, protected veteran status or disability.

The College at Brockport Tenure Track Assistant Professor

Applications are invited for a tenure track Assistant Professor position in the Department of Computing Sciences (home of ABET accredited programs) beginning Fall 2018. Doctoral degree in Computer Science, or in a closely related field is required. ABD candidates considered. Hired faculty will be expected to teach, engage in research, and participate in service appropriate to rank. Preference will be given to candidates with expertise in software security, mobile app development, software engineering, or operating systems, although candidates with expertise in any area of Computer Science will be considered. All positions are subject to final budgetary approval. Apply online at <http://www.brockportrecruit.org> by January 18, 2018. EOE/AA employer: M/F/DIS/VET

Florida State University Tenure-Track Assistant Professors

The Department of Computer Science at the Florida State University invites applications for two tenure-track Assistant Professor positions to begin August 2018. The positions are 9-month, full-time, tenure-track, and benefits eligible. We are seeking outstanding applicants with strengths in the broad areas of Data Sciences or Trustworthy Computing. While strong candidates in all related areas will be considered, the focused areas in Data Sciences are Computer Graphics, Visualization, Machine Learning, and Data Analytics; and the focused areas in

Trustworthy Computing include Formal Methods and Verification, Embedded and Cyber-Physical Systems, Digital Forensics, Compilers, and Computer Architecture. Applicants should hold a PhD in Computer Science or closely related field at the time of appointment, and have excellent research and teaching accomplishments or potential. The department offers degrees at the BS, MS, and PhD levels. The department is an NSA Center of Academic Excellence in Information Assurance Education (CAE/IAE) 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, 2017 and will continue until the positions are filled. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of three references, at: <http://www.cs.fsu.edu/positions/apply.html>

Questions can be e-mailed to Prof. Xiuwen Liu, Faculty Search Committee Chair, recruitment@cs.fsu.edu.

Requirements: PhD in Computer Science or closely related field at the time of appointment, and have excellent research and teaching accomplishments or potential.

Equal Employment Opportunity

An Equal Opportunity/Access/Affirmative Action/Pro Disabled & Veteran Employer committed



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

**Ads are listed for a period of 30 days.
For More Information Contact:**

**ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**



上海科技大学
ShanghaiTech University



TENURE-TRACK AND TENURED POSITIONS

ShanghaiTech University invites highly qualified candidates to fill multiple tenure-track/tenured faculty positions as its core founding team in the School of Information Science and Technology (SIST). We seek candidates with exceptional academic records or demonstrated strong potentials in all cutting-edge research areas of information science and technology. They must be fluent in English. English-based overseas academic training or background is highly desired.

ShanghaiTech is founded as a world-class research university for training future generations of scientists, entrepreneurs, and technical leaders. Boasting a new modern campus in Zhangjiang Hightech Park of cosmopolitan Shanghai, ShanghaiTech shall trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to both graduate and undergraduate educations.

Academic Disciplines: Candidates in all areas of information science and technology shall be considered. Our recruitment focus includes, but is not limited to: computer architecture, software engineering, database, computer security, VLSI, solid state and nano electronics, RF electronics, information and signal processing, networking, security, computational foundations, big data analytics, data mining, visualization, computer vision, bio-inspired computing systems, power electronics, power systems, machine and motor drive, power management IC as well as inter-disciplinary areas involving information science and technology.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including on-campus housing. All regular ShanghaiTech faculty members will join its new tenure-track system in accordance with international practice for progress evaluation and promotion.

Qualifications:

- Strong research productivity and demonstrated potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, Statistics, Applied Math, or related field);
- A minimum relevant (including PhD) research experience of 4 years.

Applications: Submit (in English, PDF version) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn. For more information, visit <http://sist.shanghaitech.edu.cn/NewsDetail.asp?id=373>

Deadline: The positions will be open until they are filled by appropriate candidates.

to enhancing the diversity of its faculty and students. Individuals from traditionally underrepresented groups are encouraged to apply.

FSU's Equal Opportunity Statement can be viewed at: http://www.hr.fsu.edu/PDF/Publications/diversity/EEO_Statement.pdf

Indiana University Faculty Positions in Intelligent Systems Engineering

The School of Informatics, Computing, and Engineering (SICE) at Indiana University (IU) Bloomington invites applications for multiple open rank tenured/tenure track faculty positions to begin in Fall 2018 in **Intelligent Systems Engineering (ISE)**. Duties include research, teaching, and service.

ISE is an innovative new program, currently with 19 faculty, that focuses on the intersection of sophisticated computing methods and information technology with critical engineering problems. Current foci include bioengineering, computer engineering, robotics and cyberphysical systems, molecular and nanoscale engineering, environmental engineering, neuro-engineering, and intelligent systems. ISE reflects a top priority for Indiana University, with an expected \$120 million investment and search under way for an Associate Dean for Engineering. We are particularly interested in hiring faculty whose research develops and applies advanced computational approaches, especially **intelligent systems, applied machine learning and artificial intelligence, cloud computing, cyberphysical systems,**

computer engineering, and systems security engineering to address important problems in any of these areas.

The department offers BS and Ph.D. degrees which started in fall 2016, and an MS degree has just been approved. The engineering program draws upon IU Bloomington's considerable education and research strengths such as biology, business, chemistry, computer science, environmental science, informatics, law, medicine, music, physics, network science, optometry, psychological and brain sciences, speech and hearing sciences, and statistics. New faculty will have considerable opportunity and responsibility to shape the development of curricula and research. There is a strong emphasis on world-class research, built around focused laboratories and proactively involving undergraduates. More information can be found at <https://www.engineering.indiana.edu>

Applicants should have an established record (for senior level) or demonstrable potential for excellence (for junior level) in research and teaching, and a PhD in a related field expected before August 2018.

Interested candidates should review the application requirements and submit their application at: <https://indiana.peopleadmin.com/postings/4613>

For full consideration, applications are due by January 1, 2018, but applications will be considered until the positions are filled.

Questions may be sent to isechair@indiana.edu

Indiana University is an equal employment and affirmative action employer and a provider of ADA

services. All qualified applicants will receive consideration for employment without regard to age, ethnicity, color, race, religion, sex, sexual orientation or identity, national origin, disability status or protected veteran status.

Max Planck Institute for Software Systems (MPI-SWS) Tenure-Track Openings

Applications are invited for faculty positions at all career stages in computer science, with a particular emphasis on systems (broadly construed). We expect multiple positions to be filled in systems, but exceptional candidates in other areas of computer science are also strongly encouraged to apply.

A doctoral degree in computer science or related areas and an outstanding research record (commensurate for the applicant's career stage) are required. Successful candidates are expected to build a team and pursue a highly visible research agenda, both independently and in collaboration with other groups.

MPI-SWS is part of a network of over 80 Max Planck Institutes, Germany's premier basic-research organisations. MPIs have an established record of world-class, foundational research in the sciences, technology, and the humanities. The institute offers a unique environment that combines the best aspects of a university department and a research laboratory: Faculty enjoy full academic freedom, lead a team of doctoral students and post-docs, and have the opportunity to teach university courses; at the same time, they



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Faculty Positions in Computer and Communication Sciences École polytechnique fédérale de Lausanne (EPFL)

The School of Computer and Communication Sciences (IC) at EPFL invites applications for faculty positions in computer and communication sciences. We are seeking candidates for tenure track assistant professor as well as, in exceptional cases, for senior positions.

Successful candidates will develop an independent and creative research program, will participate and be committed to excellence in undergraduate and graduate teaching, and will supervise PhD students.

The school is seeking candidates in the fields of: data science and machine learning – including applications in bioinformatics, natural language processing, and speech recognition – security and privacy, and verification and formal methods. Candidates in other areas will also be considered.

EPFL offers internationally competitive salaries, generous research support, significant start-up resources, and outstanding research infrastructure. Academics in Switzerland enjoy many research funding opportunities, as well as an exceptionally high standard of living.

To apply, please follow the application procedure at

<https://facultyrecruiting.epfl.ch/position/6848514>

The following documents are requested in PDF format: cover letter, curriculum vitae including publications list, brief statements of research and teaching interests, and contact information (name, postal address, and email) of 3 references for junior positions or 5 for senior positions. Screening will start on **December 15th, 2017**.

Further questions may be addressed to:

Prof. Anastasia Ailamaki
Chair of the Recruiting Committee
Email: recruiting.ic@epfl.ch

For additional information on EPFL and IC, please consult: <http://www.epfl.ch> and <http://ic.epfl.ch>

EPFL is an equal opportunity employer and a family friendly university.

enjoy ongoing institutional funding in addition to third-party funds, a technical infrastructure unrivaled for an academic institution, as well as internationally competitive compensation.

The institute is located in the German cities of Saarbruecken and Kaiserslautern, in the tri-border area of Germany, France, and Luxembourg. We maintain an international and diverse work environment and seek applications from outstanding researchers worldwide. The working language is English; knowledge of the German language is not required for a successful career at the institute.

Qualified candidates should apply on our application website (apply.mpi-sws.org). To receive full consideration, applications should be received by December 1st, 2017.

The institute is committed to increasing the representation of minorities, women, and individuals with physical disabilities. We particularly encourage such individuals to apply. The initial tenure-track appointment is for five years; it can be extended to seven years based on a midterm evaluation in the fourth year. A permanent contract can be awarded upon a successful tenure evaluation in the sixth year.

Mississippi State University Professor and Head Department of Computer Science and Engineering

Applications and nominations are being sought for the Professor and Head of the Department of Computer Science and Engineering (www.cse.msstate.edu) at Mississippi State University. The Head is responsible for the overall administration of the department and this is a 12-month tenured position.

The successful Head will provide vision and leadership for nationally recognized computing education and research programs; exceptional academic and administrative skills; a strong commitment to faculty recruitment and development; and a strong commitment to promoting diversity.

Applicants must have a Ph.D. in computer science, software engineering, computer engineering, or a closely related field. The successful candidate must have earned national recognition by a distinguished record of accomplishments in computer science education and research. Demonstrated administrative experience is desired, as is teaching experience at both the undergraduate and graduate levels. The successful candidate must qualify for the rank of professor.

Applicants must apply online by submitting a cover letter outlining your experience and vision for this position, a curriculum vitae, and the names and contact information of at least three professional references. The online applicant site can be accessed by going to www.msujobs.msstate.edu. Screening of candidates will begin November 1, 2017 and will continue until the position is filled. Inquiries and nominations should be directed to Dr. Nick Younan, Department Head of Electrical and Computer Engineering and Search Committee Chair (younan@ece.msstate.edu or 662-325-3912).

MSU is an equal opportunity employer, and all qualified applicants will receive consideration for employment without regard to race, color, religion, ethnicity, sex (including pregnancy and gender iden-

tity), national origin, disability status, age, sexual orientation, genetic information, protected veteran status, or any other characteristic protected by law. We always welcome nominations and applications from women, members of any minority group, and others who share our passion for building a diverse community that reflects the diversity in our student population.

Oakland University Tenure-Track Faculty Positions

The Department of Computer Science and Engineering needs to fill four tenure-track assistant professor positions. One position is in Cybersecurity area. The areas for other three positions include Human Computer Interaction, High Performance Computing, Database Systems, and Computer Networking. All positions will begin on August 15, 2018. Applicants must have completed a Ph.D. in Computer Science, Information Technology, or a closely related field by the appointment date. Candidates must show exceptional promise in both research and teaching. Candidates should have an appreciation of and commitment to the value of diversity and working with a diverse faculty and student body.

Applications should be submitted by November 30, 2017. Applicants should submit a letter of intent, a statement of research, a statement of teaching, resume, and list of three references. The candidates for cybersecurity position should upload their application at <http://jobs.oakland.edu/postings/12218>. The candidates for other three positions should upload their application at <http://jobs.oakland.edu/postings/12177>. The teaching statement should include a list of undergraduate and graduate courses that the applicant will be willing to teach as well as outlines of two courses that the applicant would like to introduce. Information about the current courses offered by the department is available on departmental website at <http://www.cse.secs.oakland.edu>.

The department is currently offering BSc. Degrees in Computer Science and in Information Technology, MSc. degrees in Computer Science, Cyber Security and in Software Engineering and Information Technology, and a Ph.D. in Computer Science and Informatics. For information about the department and Oakland University, please visit the respective homepages.

Oakland University is an ADVANCE institution, one of a limited number of universities in receipt of NSF funds in support of our commitment to increase diversity and the participation and advancement of women and underrepresented minorities in the STEM fields. Oakland University is an equal opportunity employer.

Oakland University is a nationally recognized doctoral research institution located on 1,443 acres of scenic land in the cities of Rochester Hills and Auburn Hills in Oakland County, Michigan. The University has 132 bachelor's degree programs and 138 graduate degree and certificate programs. Academics include programs in the College of Arts and Sciences, School of Business Administration, School of Education and Human Services, School of Engineering and Computer Science, School of Health Sciences, School of Medicine and School of Nursing.

Professor (Open Rank)

Looking for faculty colleagues who engage deeply in both research and teaching within a curriculum that embraces student projects and independent learning? Consider joining the faculty at WPI.

The Computer Science Department anticipates hiring multiple tenure-track faculty for the Fall of 2018 whose expertise is in the following areas:

- 1) Human-Computer Interaction;
- 2) Visualization and Visual Analytics;
- 3) Dependable Software Systems;
- 4) Algorithms, particularly with expertise that may be complementary to interdisciplinary programs in Bioinformatics, Data Science and Learning Science;
- 5) Robotics, joining faculty in our interdisciplinary Robotics Engineering program,
- 6) Interactive Media and Game Development, joining faculty in our interdisciplinary Interactive Media & Game Development program, and
- 7) Computational Neuroscience, working with faculty in Biology and our interdisciplinary Bioinformatics and Computational Biology program.

In addition to these specific areas, outstanding candidates in any area will receive full consideration. Candidates should have a PhD in Computer Science or a closely related field, and the potential for excellence in research and teaching.

WPI's reputation as a rigorous and innovative university rests on the shoulders of its faculty. A highly selective, private technological university and one of the nation's first, WPI believes that when great minds work together, great advances follow. At WPI the boundaries to multidisciplinary collaboration are low--- faculty members, students, and other partners work together on the real-world projects and purposeful research that are hallmarks of the WPI experience. We are most proud of a recent No. 1 ranking for "faculty who best combine research and teaching." (Wall Street Journal/Times Higher Ed, 2016). Located one hour west of Boston, the university's campus is in Worcester, Massachusetts, a thriving 21st century college city recognized as a growing hub of scientific and technological innovation.

Questions about the hiring process should be sent to recruit@cs.wpi.edu. More information about the positions and instructions for applying are available at <http://web.cs.wpi.edu/facultyhire/>. You will need to include detailed research and teaching statements, vitae and contact information for at least three references.

The deadline for applications is December 15, 2017 with applications continuing to be considered after that date until the positions are filled.

WPI is an Equal Opportunity Employer

GREAT MINDS at WORK



COLLEGIATE ASSISTANT PROFESSOR

Department of Computer Science

The Department of Computer Science at Virginia Tech (www.cs.vt.edu) seeks applicants for a collegiate faculty position at the Assistant Professor level. Candidates must have a Ph.D. in computer science or related field at the time of appointment. Collegiate faculty members have a primary commitment to the instructional mission of the department, including graduate and undergraduate teaching, curricular and program development, and the design and integration of innovative and inclusive pedagogy. Successful candidates should give evidence of potential to take a lead role in enhancing curricula and promoting teaching excellence. In addition to teaching, candidates will be expected to participate in research and scholarship, whether on teaching and learning or on other computer science research topics of interest. Candidates will have the opportunity to collaborate with a wide range of research groups in the department, including a thriving group in CS education research. Candidates with demonstrated knowledge of CS education research topics such as education-related software systems, analysis of student data analytics, CS education for non-majors or at the K-12 level, cybersecurity education, distance education, or diversity in CS are encouraged to apply.

The department has 47 teaching faculty including 42 tenured and tenure-track faculty, over 800 undergraduate majors, and more than 250 graduate students. The department is in the College of Engineering, whose undergraduate program ranks 14th and graduate program ranks 27th among U.S. engineering schools (*USN&WR*, 2017). The department plays a central role in several university-wide initiatives (see provost.vt.edu/destination-areas). Successful candidates will have the opportunity to participate in new transdisciplinary research programs and curricula in Data and Decisions, Integrated Security, Intelligent Infrastructure for Human-Centered Communities, and Creativity & Innovation.

The collegiate faculty rank is a non-tenure-track position that offers a clear promotion path with increasingly long-term contracts. Collegiate faculty are full members of the department faculty, and are expected to participate in sponsored research, mentor graduate students, participate in department and professional service, etc. This position is located at the main campus in Blacksburg, VA, a region consistently ranked among the country's best places to live. The position requires occasional travel to professional meetings. Successful candidates should give evidence of commitment to issues of diversity in a campus community. Virginia Tech is committed to building a culturally diverse faculty and strongly encourages applications from women and minorities. The selected candidate must pass a criminal background check prior to employment.

Applications must be submitted online to jobs.vt.edu for posting #TR0170132. Applicant screening will begin on November 27, 2017 and continue until the position is filled. Inquiries should be directed to Dr. Dennis Kafura, Search Committee Chair, kafura@cs.vt.edu.

Virginia Tech is an AA/EEO employer, committed to building a culturally diverse faculty; we strongly encourage applications from women and minorities.

Southern University of Science and Technology Multiple Tenure-Track Faculty Positions

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.

As a State-level innovative city, Shenzhen has identified innovation as the key strategy for its development. It is home to some of China's most successful high-tech companies, such as Huawei and Tencent. SUSTech considers entrepreneurship as one of the main directions of the university. SUSTech encourages candidates with experience in entrepreneurship to apply.

The Department of Computer Science and Engineering at SUSTech was founded in 2016. It has 12 professors, all of whom hold doctoral degrees or have years of experience in overseas universities. Among them, two were elected into the 1000 Talents Program in China; three are IEEE fellows; one IET fellow.

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.

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. Salary and rank will commensurate with qualifications and experience. More information can be found at <http://talent.sustc.edu.cn/en>.

We provide some of the best start-up packages in the sector to our faculty members, including one PhD studentship per year and two postdoctoral fellowships, in addition to a significant amount of start-up funding (which can be used to fund additional PhD students and postdocs, research travels, and research equipment).

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.

Texas Christian University
Assistant Professor Faculty Positions

The Department of Computer Science at Texas Christian University (TCU) invites applications for two tenure-track assistant professor positions beginning Fall 2018. Applicants should have an earned Ph.D. in Computer Science from an accredited institution, must have excellent verbal and written communication skills, and a strong commitment to both teaching and research.

Qualified applications are invited from candidates with specializations in all areas in computer science with a specialization in data analytics being preferred for one of the positions.

Applicants for either position will be expected to teach a wide variety of courses at the undergraduate level and should be willing to supervise undergraduate research projects. Responsibilities include teaching undergraduate computing courses in the Department's programs in Computer Science (COSC) and Computer Information Technology (CITE) and advising and mentoring majors; conducting research and engagement in scholarship in the applicant's area of specialization. Salary is commensurate with qualifications and experience.

TCU uses an online application protocol administered by Human Resources. All application materials: cover letter, *curriculum vitae*, representative publications, detailed research plans, a statement of teaching philosophy and interests must be submitted electronically at https://tcu.igreenetree.com/CSS_Faculty/CSSPage_Welcome.asp.

Three confidential letters of recommendation should be emailed directly from the reviewer or

dossier service to: hrtalentacquisition@tcu.edu. Do not send documents directly to the Department.

Review of applications will begin immediately and continue until both positions are filled.

TCU (www.tcu.edu) is a private, coeducational university within easy reach of many centers of business and research located in the Dallas-Fort Worth Metroplex (DFW). As an AA/EEO employer, TCU recruits, hires, and promotes qualified persons in all job classifications without regard to age, race, color, religion, sex, sexual orientation, gender, gender identity, gender expression, national origin, ethnic origin, disability, genetic information, covered veteran status, or any other basis protected by law.

Texas State University
Department of Computer Science

The Department of Computer Science invites applications for three faculty positions:

1. Two tenure-track Assistant Professor positions to start on September 1, 2018. Review date is January 8, 2018. We are seeking candidates to complement and enhance our research in data analytics, human-computer interactions, artificial intelligence, computer security and networks, high-performance computing and software engineering. Outstanding candidates in other areas will also be considered. Job duties include conducting research that results in refereed publications and external funding, teaching effectively at the graduate and undergraduate levels, supervising student research, and serving

at the department, college, university, and professional levels.

2. One non-tenure track Senior Lecturer position to start on January 16, 2018. Review date is October 15, 2017. The candidate is expected to teach a variety of courses primarily at the undergraduate level and serve at the department, college, and university levels. This non-tenure-line, nine-month faculty position will have a contract term not to exceed five years, subject to annual re-appointment review, and renewable upon expiration of the initial term.

As a non-tenure line faculty member, the candidate is not expected to engage in research, external funding, or publications. However, collaborative participation with colleagues in such activities as well as service, curriculum initiatives, and conducting learning outcomes assessment, can be considered in the annual evaluation.

Consult the department's page at www.cs.txstate.edu/employment/faculty/ for required and preferred qualifications, application procedures, and information about the university and the department.

Texas State University is committed to an inclusive education and work environment that provides equal opportunity and access to all qualified persons. Texas State, to the extent not in conflict with federal or state law, prohibits discrimination or harassment on the basis of race, color, national origin, age, sex, religion, disability, veterans' status, sexual orientation, gender identity or expression. Texas State University is a member of The Texas State University System. Texas State University is an EOE.

Carnegie Mellon University
Africa

CARNEGIE MELLON UNIVERSITY
COLLEGE OF ENGINEERING | FACULTY POSITIONS IN ELECTRICAL AND COMPUTER ENGINEERING IN AFRICA

The College of Engineering at Carnegie Mellon University, a world leader in information and communication technology, has extended its global reach into Africa. Offering master's degrees to full-time resident students from across Africa at our base in Kigali, Rwanda, CMU-Africa is educating future leaders who will create the technology and business innovations that will transform Africa.

We are seeking highly qualified faculty candidates at all levels, from new PhDs to senior personnel, to join our dynamic, world-class faculty in contributing to the emerging knowledge-based economies across the continent. CMU-Africa faculty members collaborate with industry and deliver innovative, interdisciplinary graduate teaching and research programs in the African context.

Carnegie Mellon is seeking exceptional candidates who can deliver innovative, interdisciplinary graduate programs in the following areas:

- Software engineering
- Mobile and cloud computing
- Communications and wireless networking
- Cyber-security and privacy
- Embedded systems
- Internet of things
- Energy systems
- Data analytics
- Machine learning
- Applications in healthcare, agriculture, finance and infrastructure
- Innovation, entrepreneurship and technology management

Candidates should possess a Ph.D. from a leading research university with a serious interest in both teaching and research in the context of opportunities in Africa. We are particularly interested in applicants who have passion for a culturally diverse environment and who demonstrate a willingness to nurture the inclusive Carnegie Mellon environment. We are actively committed to considering a diverse applicant pool in terms of gender, race, veteran status, and disability. Carnegie Mellon University seeks to meet the needs of dual-career couples and is a member of the Higher Education Recruitment Consortium (HERC) that assists with dual-career searches.

Applications should include a comprehensive resume including a complete list of publications, 3-5 professional references, a statement of research and teaching interests (less than 2 pages each), and copies of 2 research papers (journal or conference papers). We invite you to learn more about our program and the exciting opportunities in the new Africa at www.cmu.edu/africa.

APPLICATIONS SHOULD BE SENT TO: Director, Carnegie Mellon University-Africa, email: director@africa.cmu.edu

ACM Transactions on Spatial Algorithms and Systems



ACM TSAS is a new scholarly journal that publishes high-quality papers on all aspects of spatial algorithms and systems and closely related disciplines. It has a multi-disciplinary perspective spanning a large number of areas where spatial data is manipulated or visualized.

The journal is committed to the timely dissemination of research results in the area of spatial algorithms and systems.



For further information
or to submit your
manuscript,
visit tsas.acm.org

U.S. Naval Academy Distinguished Visiting Professors

The U.S. Naval Academy's Computer Science Department invites applications for one or more Distinguished Visiting Professors. The visiting professor is expected to have a strong reputation and technical expertise in Computer Science, Information Technology, or a closely related field.

The start date of this position is flexible but August 2018 is preferred. We have provisions for either full-time financial support or supplemental support for a professor on sabbatical. The position duration could vary from one to several years. Responsibilities may vary and may include teaching, collaborating with faculty and/or mentoring student research.

The Computer Science Department offers majors in Computer Science and Information Technology, and contributes to a new major in Cyber Operations. The department is housed in a state of the art building overlooking the scenic Severn River. Our spaces provide outstanding office, laboratory, and research facilities for both students and faculty.

The Naval Academy is an undergraduate institution located in historic downtown Annapolis, Maryland on the Chesapeake Bay. Over half of the faculty are tenured or tenure track civilian professors with Ph.D.s who balance teaching excellence with internationally recognized research programs. The remaining faculty are active duty military officers with Masters or Doctoral degrees. Each year the academy graduates roughly 1000 undergraduate students with majors in the sciences, engineering, and humanities. More information about the department and the Academy can be found at <http://www.usna.edu/cs/> and <http://www.usna.edu/>.

For more information on the position, and to apply, go to <https://www.usna.edu/HRO/jobinfo/DistinguishedVisitingProf-CompSci.php>.

The University of Alabama in Huntsville Assistant Professor

The Department of Computer Science at The University of Alabama in Huntsville (UAH) invites applicants for a tenure-track faculty position at the Assistant Professor level beginning August 2018 in the area of cybersecurity; however, outstanding candidates in other areas such as cloud computing and mobile computing may be considered if they are qualified to teach undergraduate and graduate courses in cybersecurity.

A Ph.D. in computer science or a closely related area is required. The successful candidate will have a strong academic background and be able to secure and perform funded research in areas typical for publication in well-regarded academic conference and journal venues. In addition, the candidate should embrace the opportunity to provide undergraduate education.

The department has a strong commitment to excellence in teaching, research, and service; the candidate should have good communication skills, strong teaching potential, and research accomplishments.

UAH is located in an expanding, high technology area, in close proximity to Cummings Research Park, the second largest research park in the nation and the fourth largest in the world. Nearby are

the NASA Marshall Space Flight Center, the Army's Redstone Arsenal, Fortune 500 companies, and numerous high tech enterprises. UAH also has an array of research centers, including information technology and cybersecurity. In short, collaborative research opportunities are abundant, and many well-educated and highly technically skilled people are in the area. There is also access to excellent public schools and inexpensive housing.

UAH has an enrollment of approximately 9,100 students. The Computer Science department offers BS, MS, and PhD degrees in Computer Science. Approximately 554 undergraduate majors and 151 graduate students are associated with the unit. Faculty research interests are many and include cybersecurity, mobile computing, data science, software engineering, visualization, graphics and game computing, multimedia, AI, image processing, pattern recognition, and distributed systems. Recent NSF figures indicate the department ranks 30th in the nation in overall federal research funding.

Interested parties must submit a detailed resume with references to info@cs.uah.edu or Chair, Search Committee, Dept. of Computer Science The University of Alabama in Huntsville, Huntsville, AL 35899. Qualified female and minority candidates are encouraged to apply. Initial review of applicants will begin as they are received and continue until a suitable candidate is found.

The University of Alabama in Huntsville is an affirmative action/equal opportunity employer/minorities/females/veterans/disabled.

Please refer to log number: 18/19-530

University of California, Irvine Donald Bren School of Information and Computer Sciences Professor and Professor of Teaching Series Positions

The Donald Bren School of Information and Computer Sciences (ICS) at the University of California, Irvine (UCI) is seeking exceptional candidates for multiple tenured/tenure-track positions in the Professor and Professor of Teaching series.

Professor Series

*Cybersecurity
Data Science
Computer Systems
Human-Computer Interaction
Software Engineering*

Professor of Teaching Series

*Computer Science
Informatics
Statistics*

Professor of Teaching positions are full-time faculty positions with an emphasis on teaching that parallel the Professor series and include an expectation of research and service contributing to instruction and pedagogy. They are designed for individuals who wish to focus their careers on teaching, professional activities, and University and public service. A detailed description of each position, and application instructions, can be found on the ICS website at <http://ics.uci.edu>.

ICS is comprised of three departments (Computer Science, Informatics, and Statistics), and it is one of only five computing-focused schools among the Association of American Universities

(AAU) members. The U.S. News and World Report 2017 Best Global Universities ranking identifies UCI as a top 50 university in computer science and one of the top 15 universities for computer science in the United States. The School's 70+ faculty members include 1 NAE Member, 14 ACM Fellows, 9 IEEE Fellows, 7 AAAS Fellows and many other national award winners.

The University of California, Irvine is ranked as a top ten public university by U.S. News and World Report, and has been identified by the New York Times as No. 1 among U.S. universities that do the most for low-income students. UCI has done what no other school has done—rank among Sierra's Top 10 most sustainable colleges for eight years in a row. UCI is located in Orange County, 4 miles from the Pacific Ocean and 45 miles south of Los Angeles. Irvine is one of the safest communities in the U.S. and offers a very pleasant year-round climate, numerous recreational and cultural opportunities, and one of the highest-ranked public school systems in the nation.

The University of California, Irvine is an Equal Opportunity/Affirmative Action Employer advancing inclusive excellence. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, disability, age, protected veteran status, or other protected categories covered by the UC nondiscrimination policy.

University of California, Riverside Assistant Teaching Professor

The Department of Computer Science and Engineering (CSE) at the University of California, Riverside invites applications for an Assistant Teaching Professor position beginning in January 2018. At UCR, teaching professors are responsible mainly for undergraduate instruction and curriculum development. Successful candidates will need to exhibit dedication to teaching and appropriate pedagogical knowledge and skills. Priority will be given to candidates with expertise and prior experience in teaching courses on data structures, software engineering, discrete mathematics, and algorithms, but highly qualified candidates with background in other areas will also be given consideration. Besides teaching, teaching professors are also expected to be actively engaged in development of undergraduate curricula, pedagogical innovation, TA training, program accreditation, student advising, and scholarly activity in the area of computer science education.

An assistant teaching professor appointment is similar to a regular tenure-track assistant professor appointment and follows a parallel track, including regular, rigorous performance reviews and a tenure process that, if successful, leads to a tenured appointment.

A Ph.D. in Computer Science or a related field is required at the time of employment. Salary will be competitive and commensurate with qualifications and experience. Full consideration will be given to applications received by October 30, 2017. The search will continue until the position is filled.

To apply, please register through the weblink at <https://aprecruit.ucr.edu/apply/JPF00798> or at <http://www.engr.ucr.edu/about/employment.html>. Inquiries should be directed to lsosresearch@cs.ucr.edu.

UCR is a world-class research university with an exceptionally diverse undergraduate student body. Its mission is explicitly linked to providing routes to educational success for underrepresented and first-generation college students. A commitment to this mission is a preferred qualification.

The University of California, Riverside is an Equal Opportunity/Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, age, disability, protected veteran status, or any other characteristic protected by law.

EEO/AA/ADA/Vets Employer

University of California, Riverside Assistant Teaching Professor

The Department of Computer Science and Engineering (CSE) at the University of California, Riverside invites applications for an Assistant Teaching Professor position beginning in July 2018. At UCR, teaching professors are responsible mainly for undergraduate instruction and curriculum development. Successful candidates will need to exhibit dedication to teaching and appropriate pedagogical knowledge and skills. Priority will be given to candidates with expertise and prior experience in teaching courses on topics related to computer organization and architecture, logic design, and high-level synthesis. Highly qualified candidates with background in other areas will also be given consideration. Besides teaching, teaching professors are also expected to be actively engaged in service (e.g., development of undergraduate curricula, pedagogical innovation, TA training, program accreditation, student advising) and scholarly activity in the area of computer science education.

An assistant teaching professor appointment is similar to a regular tenure-track assistant professor appointment and follows a parallel track, including a tenure process that, if successful, leads to a tenured appointment. Advancement through the faculty ranks at the University of California is through a series of structured, merit-based evaluations, occurring every 2-3 years, each of which includes substantial peer input.

A Ph.D. in Computer Science or a related field is required at the time of employment. Salary will be competitive and commensurate with qualifications and experience. Full consideration will be given to applications received by January 30, 2018. The search will continue until the position is filled. To apply, please register through the weblink at <https://aprecruit.ucr.edu/apply/JPF00853> or <http://www.engr.ucr.edu/about/employment.html>. Inquiries should be directed to lsosresearch@cs.ucr.edu.

UCR is a world-class research university with an exceptionally diverse undergraduate student body. Its mission is explicitly linked to providing routes to educational success for underrepresented and first-generation college students. A commitment to this mission is a preferred qualification.

The University of California, Riverside is an Equal Opportunity/Affirmative Action Employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity,

national origin, age, disability, protected veteran status, or any other characteristic protected by law.

University of Illinois at Chicago Department of Computer Science Open-Rank Tenure-Track Faculty Positions

Located in the heart of Chicago, the Computer Science Department at the University of Illinois at Chicago (UIC) invites applications for many full-time tenure-track positions at all ranks. All candidates must have a doctorate in Computer Science or a closely related field by the appointment's starting date. Candidates will be expected to demonstrate excellence in research and teach effectively at the undergraduate and graduate levels.

We seek candidates in **all areas** of computing, at **all levels**, with special but not exclusive interest in fields related to speech and/or natural language processing, computer vision, programming languages and compilers, machine learning, human-computer interaction, data science, and computer systems. Over the next few years, we expect to hire multiple faculty in all of those areas and many others. Applicants working at the intersection of computer science and related disciplines are also encouraged to apply.

Applications must be submitted at <https://jobs.uic.edu/>, and must include a curriculum vitae, teaching and research statements, and names and addresses of at least three references in the online application. Links to a professional website and Google Scholar, ResearchGate, or similar profiles are recommended, but not required. Applicants may contact the Faculty Search Chair at search-chair@cs.uic.edu for additional information. For fullest consideration, apply by November 15, 2017. Applications will be accepted until the positions are filled.

The rapidly growing department of Computer Science at UIC has 33 tenure-system faculty - 13 of whom are NSF CAREER award recipients - with strong and broad research agendas. The department is committed to building a diverse faculty preeminent in its missions of research, teaching, and service to the community. Candidates who have experience engaging with a diverse range of faculty, staff, and students, and contributing to a climate of inclusivity are encouraged to discuss their perspectives on these subjects in their application materials.

UIC is a major public research university (R1, according to the Carnegie Classification of Institutions of Higher Education) with over 1,900 faculty and over 30,000 students. UIC is committed to increasing access to education, employment, programs and services for all. The University of Illinois is an Equal Opportunity, Affirmative Action employer. Minorities, women, veterans, and individuals with disabilities are encouraged to apply. UIC is responsive to the concerns of dual-career couples.

Chicago epitomizes the modern, livable, vibrant, and diverse city. World-class amenities like the lakefront, arts and culture venues, festivals, and two international airports make Chicago a singularly enjoyable place to live. Yet the cost of living, whether in an 88th floor condominium downtown or on a tree-lined street in one of the nation's finest school districts, is remarkably affordable.

University of Maryland, Baltimore County

An Honors University in Maryland Tenure Track position in Health Information Technology (Health IT)

The Department of Information Systems (IS) at UMBC invites applications for a **tenure-track** faculty position at the Assistant Professor level starting August 2018. We are searching for candidates with research interests and experience in Health IT, a research area with high growth and impact in healthcare and related fields. The ideal candidate will have expertise in conducting research that impacts healthcare outcomes, quality, and costs, and intersects with active research areas in the IS department: Artificial Intelligence/ Knowledge Management, Data Science, Human Centered Computing, Software Engineering, and Health Information Technology. Strong candidates with research emphases in other areas may also be considered. Candidates must have earned a PhD in Information Systems or a related field no later than August 2018.

Preference will be given to those who can collaborate with current faculty within and across departments at UMBC, fostering interdisciplinary research. Candidates are expected to establish a collaborative, externally funded, and nationally recognized research program as well as contribute to graduate and undergraduate teaching, advising, and mentoring. We especially welcome applications from candidates who are willing to contribute to the diversity mission of the university. The IS Department is committed to increasing the diversity of our community. The Department offers undergraduate degrees in Information Systems and Business Technology Administration. Graduate degree programs, MS and PhD, are offered in both Information Systems and Human-Centered Computing, including an innovative online MS program in IS ranked among the top 20 in the nation by US News & World Report. Consistent with UMBC's vision and mission, the Department has excellent teaching facilities, state-of-the-art laboratories, and outstanding technical support. Further details on our research, academic programs, and faculty can be found at <http://www.is.umbc.edu>.

UMBC is a dynamic public research university integrating teaching, research, and service. As an Honors University, the campus offers academically talented students a strong undergraduate liberal arts foundation that prepares them for graduate and professional study, entry into the workforce, and community service and leadership. UMBC emphasizes science, engineering, information technology, human services, and public policy at the graduate level. UMBC contributes to the economic development of the State and the region through health equity, entrepreneurial initiatives, workforce training, K-16 partnerships, and technology commercialization in collaboration with public agencies and the corporate community. Diversity is a core value of UMBC and we believe that the educational environment is enhanced when diverse groups of people with diverse ideas come together to learn. Therefore, members of underrepresented groups including women, minorities, veterans, and individuals with disabilities are especially encouraged to apply.

UMBC continues to be recognized in *U.S. News and World Report's* national university rankings,

placing seventh in Most Innovative National Universities and 13th in Best Undergraduate Teaching. *Princeton Review* features UMBC as one of the nation's top universities, and one of the Colleges that Pay You Back. The *Chronicle of Higher Education* has listed UMBC in the "honor roll" of "Great Colleges to Work For" for eight consecutive years; it is the only Maryland four-year institution to be so recognized. Our strategic location in the Baltimore-Washington corridor puts us close to many important federal laboratories, agencies, and high-tech companies. UMBC's campus is located on 500 acres just off I-95 between Baltimore and Washington DC, less than 10 minutes from the BWI airport and Amtrak station. The campus includes a center for entrepreneurship, and the bwtech@UMBC research and technology park, which has special programs for startups focused on cybersecurity, clean energy, life sciences, and training. We are surrounded by one of the greatest concentrations of commercial, cultural, and scientific activity in the nation. Located at the head of the Chesapeake Bay, Baltimore has all the advantages of modern, urban living, including professional sports, major art galleries, theaters, and a symphony orchestra. The city's famous Inner Harbor area is an exciting center for entertainment and commerce. The nation's capital, Washington, DC, is a great tourist attraction with its historical monuments and museums. Just ten minutes from downtown Baltimore and 30 minutes from the D.C. Beltway, UMBC offers easy access to the region's resources by car or public transportation.

Electronic submission of application is required at <http://apply.interfolio.com/45260>. All applications must be submitted as PDF files, including a cover letter, CV, one-page statement of teaching interests, one-page statement of research interests, and names and contact information of at least three references. For inquiries, please contact Dr. Aryya Gangopadhyay at (410) 455-2620 or gangopad@umbc.edu. Review of applications will begin in November 2017 and will continue until the position is filled, subject to the availability of funds.

UMBC is an Affirmative Action/Equal Opportunity Employer and welcomes applications from minorities, women, veterans, and individuals with disabilities.

University of Maryland, Baltimore County

Computer Science and Electrical Engineering Tenure-Track Assistant Professor

UMBC's Department of Computer Science and Electrical Engineering invites applications for a tenure-track Assistant Professor position to begin in Fall 2018. Exceptionally strong candidates for higher ranks may be considered. Applicants must have or be completing a Ph.D. in a relevant discipline, have demonstrated the ability to pursue a research program, and have a strong commitment to undergraduate and graduate teaching.

We welcome candidates in all areas of specialization. Some areas of particular interest include, (but are not limited to): information assurance and cybersecurity; mobile, wearable, and IoT systems; big data with an emphasis on machine learning, data science, and high-performance computing; knowledge and database systems, and visualization.

The CSEE department is energetic, research-oriented and multi-disciplinary with programs in Computer Science, Computer Engineering, Electrical Engineering and Cybersecurity. Our faculty (35 tenure-track, 11 teaching and 17 research) enjoy collaboration, working across our specializations as well as with colleagues from other STEM, humanities and the arts departments and external partners. We have 1650 undergraduate and 500 M.S. and Ph.D. students in our programs.

UMBC is a dynamic public research university integrating teaching, research and service. The 2018 US News and World Report Best Colleges report placed UMBC 7th in the Most Innovative National Universities category and 13th in Best Undergraduate Teaching, National Universities. Our strategic location in the Baltimore-Washington corridor is close to many federal laboratories and agencies and high-tech companies, facilitating interactions, collaboration, and opportunities for sabbaticals and visiting appointments.

Applicants should submit a cover letter, statement of teaching and research experience and interests, CV, and three letters of recommendation at <http://apply.interfolio.com/45784>. Applications received by December 15, 2017 are assured full consideration. Send questions to jobsTT@csee.umbc.edu and see <http://csee.umbc.edu/jobs> for more information. UMBC is an affirmative action/equal opportunity employer.

University of Nevada, Las Vegas Big Data/Health Disparities, Associate/Full Professor

The University of Nevada, Las Vegas invites applications for Big Data/Health Disparities, Associate/Full Professor [18533]

PROFILE of the UNIVERSITY

UNLV is a doctoral-degree-granting institution of approximately 29,000 students and more than 3,000 faculty and staff that is classified by the Carnegie Foundation for the Advancement of Teaching as a research university with high research activity. For more information, visit us on line at: <http://www.unlv.edu>

COMMITMENT to DIVERSITY

The successful candidate will demonstrate support for diversity, equity and inclusiveness as well as participate in maintaining a respectful, positive work environment.

ROLE of the POSITION

The Department of Computer Science at the University of Nevada, Las Vegas (UNLV) invites applications for a full-time tenure-track/tenured, Associate/Full Professor position in Big Data commencing Fall 2018. A distinguished record in undergraduate and graduate computer science education, well-funded scholarly research, and service qualifications to the Big Data community will be required for appointment as a tenured Associate/Full Professor.

QUALIFICATIONS

This position requires a Ph.D. in Computer Science from a regionally accredited college or university. The applicants are expected to have an extensive research and funding in Big Data applications in health/bioinformatics. In addition, the applicants should have established research

records in one or more areas of Big Data including data mining, data analytics, data visualization, database modeling, machine learning, scalable computing, software and hardware systems for big data processing, and distributed/parallel computing.

SALARY RANGE

Salary competitive with those at similarly situated institutions. Position is contingent upon funding.

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 this position will remain open until filled, review of candidates' materials will begin on January 16, 2018 and best consideration will be gained for materials submitted prior to that date. Materials should be addressed to Dr. Kazem Taghva, Search Committee Chair, and are to be submitted via on-line application at <https://hrsearch.unlv.edu>. For assistance with UNLV's on-line applicant portal, contact UNLV Employment Services at (702) 895-3504 or applicant.inquiry@unlv.edu.

For further inquiries, please contact Dr. Kazem Taghva at Kazem.Taghva@unlv.edu
EEO/AA/Vet/Disability Employer

University of Notre Dame Special Professional Faculty

The Department of Computer Science and Engineering at the University of Notre Dame seeks candidates for a full-time (2-3 courses per semester) Special Professional Faculty (SPF) position to teach courses primarily in the CSE undergraduate curricula. Initial appointment will be made for a term of three years at the Assistant Teaching Professor, Associate Teaching Professor, or Teaching Professor level depending on seniority and experience. Appointments are renewable for five-year terms and promotions to more senior ranks are available, depending on performance.

Competitive candidates will have the training and experience necessary to teach effectively in a range of courses in accredited degree programs in Computer Science and Computer Engineering. Candidates with backgrounds in all areas of Computer Science and Computer Engineering will be considered and relevant industry experience is also valued. Qualified candidates should have at least a master's degree, and preferably a doctoral degree, in Computer Science, Computer Engineering, or a related area. The Department is especially interested in candidates who will contribute to the diversity and excellence of the University's academic community through their teaching and service.

The University of Notre Dame is a private, Catholic university with a doctoral research extensive Carnegie classification, and consistently

ranks in USNWR as a top-twenty national university. The CSE Department offers the Ph.D. degree and undergraduate Computer Science and Computer Engineering degrees. More information about the department can be found at: <http://cse.nd.edu/>

Applicants must submit a CV, cover letter, statement of teaching experience and philosophy, and contact information for three professional references, at least two of whom must be able to comment on the applicant's teaching experience. Teaching evaluations may be submitted, if available. Applications must be submitted at <http://apply.interfolio.com/45448>.

To guarantee full consideration, applications must be received by January 1, 2018, however, review of applications will continue until the position has been filled.

The University is an Equal Opportunity and Affirmative Action employer; we strongly encourage applications from women, minorities, veterans, individuals with a disability and those candidates attracted to a university with a Catholic identity.

University of Rochester Faculty Positions in Computer Science

The Computer Science Department at the University of Rochester seeks applicants for two tenure-track positions. Outstanding candidates will be considered in any area of computer science and at any level of seniority. We are particularly eager to grow our strength in human-computer interaction and in the theory and practice of security and privacy.

Candidates must have (or be about to receive) a doctorate in computer science or a related discipline. Applications should be submitted online (at <https://www.rochester.edu/faculty-recruiting/login>) no later than January 1, 2018, for full consideration; submissions beyond this date risk being overlooked due to limited interview slots.

The Department of Computer Science (<https://www.cs.rochester.edu>) has a distinguished history of research in artificial intelligence, HCI, systems, and theory. We nurture a highly collaborative and interdisciplinary culture, with exceptionally strong external funding and with active ties to numerous allied departments, including brain and cognitive science, linguistics, biomedical engineering, electrical and computer engineering, and several departments in the medical center. Recent faculty hires have received a host of national honors, including the NSF CAREER award, the MIT TR35 award, honorable mention in the ACM dissertation competition, multiple Google research awards, and best paper designations at top-tier conferences. In 2015 we were one of only two CS departments nationwide to secure three NSF CRII awards for junior faculty.

The department is deeply committed to building a more diverse and representative faculty, and strongly encourages applications from groups underrepresented in higher education. We have a vibrant Women in Computing community, and are a charter member of the ABI/HMC BRAID Initiative. With funding from the NSF, the CRA, and major industrial sponsors, BRAID works to increase diversity and inclusivity in the undergraduate program and to rigorously evaluate fac-

tors that contribute to change. In 2017, women constituted 33% of our BA/BS graduates, and we are actively working to improve the environment for other underrepresented groups.

The University of Rochester is a private, Tier I research institution with approximately 5,000 undergraduates and a comparable number of graduate students. It has recently committed \$50M to the multidisciplinary Goergen Institute for Data Science (GIDS), of which Computer Science is the leading departmental member — and with which it shares a newly constructed state-of-the-art facility. Ongoing hiring in GIDS provides exciting opportunities for collaboration between computing and other disciplines.

Anchoring the Finger Lakes region of western New York State, the greater Rochester area is home to over a million people, and offers unsurpassed quality of life, with a thriving arts scene, outstanding public schools, affordable housing, and a huge range of cultural and recreational opportunities. Traditionally strong in optics research and manufacturing, the area was recently selected by the Department of Defense as the hub of a \$600M Integrated Photonics Institute for Manufacturing Innovation.

The University of Rochester, an Equal Opportunity Employer, has a strong commitment to diversity and actively encourages applications from candidates from groups underrepresented in higher education.

*EEO Minorities/Females/
Protected Veterans/Disabled*

University of South Carolina Faculty Position in Cybersecurity

The University of South Carolina invites applications for a tenure-track faculty position at open rank in the Department of Computer Science and Engineering (<http://www.cse.sc.edu>) starting Fall 2018. The department will consider exceptional candidates in any cybersecurity areas, but is particularly interested in candidates whose primary research expertise is in trustworthy embedded systems; secure wireless networks; safety, security, & reliability of cyber-physical systems and Internet of Things.

Applicants should possess a Ph.D. degree in computer science, computer engineering, or a closely-related field, and a demonstrated record of research accomplishments in the area of cybersecurity. Prior teaching experience is preferable, but not required. The successful candidate will be expected to develop internationally-recognized, externally-funded research programs that complement existing strengths in the College and University, and to participate in college-wide, cross-cutting projects. For details on these initiatives, please visit: <http://cec.sc.edu/employment>.

USC is designated by the National Security Agency (NSA) and the Department of Homeland Security (DHS) as a National Center of Academic Excellence in Information Assurance and Cyber Defense Education and Research. Cybersecurity education and research activities are centered in the Department of Computer Science and Engineering in the College of Engineering and Computing. The Department aims to build and expand on its ongoing initiatives to develop a strong, federally-funded cybersecurity research

center. Applicants with existing or pending projects funded by NSA, DHS, or other federal agencies are encouraged to apply.

The Department of Computer Science and Engineering offers B.S. degrees in Computer Science, in Computer Information Systems, and in Computer Engineering; M.S. and Ph.D. degrees in Computer Science and in Computer Engineering; M.S. degrees in Software Engineering and in Information Security; and a Graduate Certificate in Cyber Security Studies. The Department has 23 full-time faculty members (11 of whom are NSF CAREER Award recipients), an undergraduate enrollment of 921 students, and a graduate enrollment of 168 students.

Review of applications will begin on November 1, 2017 and continue until positions are filled. Expected start date is August 16, 2018. Interested applicants will apply online at <http://cec.sc.edu/jobs/CYBER> with: (1) a letter of intent, (2) curriculum vitae, (3) a concise description of research plans, (4) a teaching plan, and (5) names & contact information of 3-5 references.

Questions may be directed to:

Department of Computer Science and Engineering

Dr. John Rose, Professor and Faculty Search

Committee Chair

rose@cse.sc.edu

(803) 777-2405

The University of South Carolina is an affirmative action, equal opportunity employer, and does not discriminate in educational or employment opportunities or decisions for qualified persons on the basis of race, sex, gender, age, color, religion, national origin, disability, genetics, sexual orientation or veteran status. Minorities and women are encouraged to apply. USC is responsive to the needs of dual career couples.

University of South Carolina

Multiple Open-Rank, Tenured or Tenure-Track Faculty Positions

The College of Engineering and Computing at the University of South Carolina is in the process of expanding its tenured and tenure-track ranks by over 40 faculty members. As part of this growth, the Department of Computer Science and Engineering (<http://cse.sc.edu>) seeks dynamic new tenured and tenure-track faculty members (at all ranks) for Fall 2018. Applicants should possess a Ph.D. degree in computer science, computer engineering, or a closely-related field, and a demonstrated record of research accomplishments. The successful candidate will be expected to develop internationally-recognized, externally-funded research programs that complement existing departmental strengths. We also desire candidates whose expertise aligns with vital cross-cutting initiatives identified by the College. For details on these initiatives, please visit: <http://cec.sc.edu/employment>.

In alignment with these initiatives, research themes of particular interest to the Department are those related to Smart & Connected Communities, Transformational Computing, Healthcare Transformation, and Smart & Agile Manufacturing. Areas of special interest include: smart technologies, deep learning, adaptive & resilient cybersecurity, computational science methods,

high-performance computing, computer architecture, artificial intelligence, cyber-physical systems, wireless networking, mobile computing, software engineering, and human-computer interaction.

The Department of Computer Science and Engineering offers B.S. degrees in Computer Science, in Computer Information Systems, and in Computer Engineering; M.S. and Ph.D. degrees in Computer Science and in Computer Engineering; M.S. degrees in Software Engineering and in Information Security; and a Graduate Certificate in Cyber Security Studies. The Department has 23 full-time faculty members (11 of whom are NSF CAREER Award recipients), an undergraduate enrollment of 921 students, and a graduate enrollment of 168 students.

Review of applications will begin on November 1, 2017 and continue until positions are filled. Expected start date is August 16, 2018. Interested applicants will apply online at <http://cec.sc.edu/jobs/CSE> with: (1) a letter of intent, (2) curriculum vitae, (3) a concise description of research plans, (4) a teaching plan, and (5) names & contact information of 3-5 references.

Questions about the departmental search may be directed to:

Department of Computer Science and Engineering

Dr. John Rose, Professor and Faculty Search

Committee Chair

rose@cse.sc.edu

(803) 777-2405

The University of South Carolina is an affirmative action, equal opportunity employer, and does not discriminate in educational or employment opportunities or decisions for qualified persons on the basis of race, sex, gender, age, color, religion, national origin, disability, genetics, sexual orientation or veteran status. Minorities and women are encouraged to apply. USC is responsive to the needs of dual career couples.

University of South Carolina Lancaster

Assistant Professor or Instructor of Computer Science

The University of South Carolina Lancaster, a Palmetto College campus of the University of South Carolina located approximately thirty-five (35) miles south of Charlotte, NC seeks candidates for a faculty position in Computer Science beginning August 16, 2018.

Applicants who hold the Ph.D. in Computer Science including credentials necessary for departmental approval to teach introductory computer science courses, as well as upper-division computer science courses as needed may be considered for a tenure-track appointment at the rank of Assistant Professor. An Assistant Professor will normally teach twelve credit hours per semester or its equivalent, including evening and distributed learning courses through Palmetto College as needed, as well as a commitment to scholarship and university/community service.

Applicants who hold a Master of Science or other appropriate degree in computer science or a related field, including credentials necessary for departmental approval to teach introductory computer science courses, as well as upper-division

computer science courses as needed may be considered for a non-tenure-track appointment at the rank of Instructor. An Instructor will normally teach twelve credit hours per semester or its equivalent, including evening and distributed learning courses through Palmetto College as needed, as well as a commitment to university/community service.

Preference will be given to candidates with experience teaching at the university level. This experience could have been gained during the course of pursuing a graduate degree.

Application materials are accepted online <https://uscjobs.sc.edu/> (Posting Number FAC00122). Applicants must complete an application, upload a cover letter that specifically addresses the applicant's qualifications, provide a curriculum vitae, and share graduate transcripts online. Applicants must also provide contact information, to include an email address, for a minimum of three professional references. Only applications and materials submitted through the online application process will be considered.

The University of South Carolina Lancaster, through the State of South Carolina and Public Employee Benefit Authority (PEBA), offers state employees a valuable benefits package, including health and life insurance, generous paid leave and retirement programs.

The University of South Carolina Lancaster does not discriminate in educational or employment opportunities on the basis of race, color, religion, national origin, sex, sexual orientation, gender, age, disability, veteran status or genetics.

The University of Texas at San Antonio

Faculty Position in Computer Science

The Department of Computer Science at The University of Texas at San Antonio (UTSA) invites applications for two tenured/tenure-track positions, starting in Fall 2018. The first position is for a tenure-track Assistant or tenured/tenure-track Associate Professor in Game-related areas. The focus is on Computer Graphics, especially 3D animation, 3D modeling, and real-time rendering; and/or Human Computer Interaction, especially human computer interfaces, virtual reality, augmented reality, and game analytics. The second position is for a tenured/tenure-track Associate Professor in Data Science and Artificial Intelligence, focusing on cyber security, Internet of things, bioinformatics, natural language processing, speech recognition, language understanding, computer vision, or machine learning. This position is part of UTSA's focused cluster hiring plan under the Gold Star Initiative to recruit top-tier researchers over a four-year period.

See <http://www.cs.utsa.edu/fsearch> for information on the Department and application instructions. Screening of applications will begin immediately. The search will continue until the positions are filled or the search is closed.

The University of Texas at San Antonio is an Affirmative Action/Equal Opportunity Employer.

Department of Computer Science

RE: Faculty Search

The University of Texas at San Antonio

One UTSA Circle

San Antonio, TX 78249-0667

Phone: 210-458-4436

[CONTINUED FROM P. 112] **challenges interdisciplinary education presents.**

I think the solution is not to say, 'we need to teach everybody everything.' It simply isn't going to work. Instead, I sometimes say that we're trying to educate people who will be 'T-shaped,' so they'll have depth in a field—that's the vertical bar of the T. But they'll also be able to engage with other people, perhaps to learn some part of a new vocabulary from other fields, so at least the conversation will be on common ground.

These days, it's difficult enough to keep up with all the changes in a single field, let alone across multiple disciplines.

Certainly now, in the field of computing, you have to. When I was a graduate student, I felt like I could go to anybody's job interview talk or thesis defense and understand what was going on. But the field has grown so much now that it's hard for a systems person to understand a thesis in theory, or for an AI person to understand a systems thesis. You have to assume that computer science will continue to grow and transform, and so, you have to be willing to learn new things.

What made you inclined to get involved with the administrative side of academia?

I'm probably the frog in the proverbial pot of water, where you raise the temperature slowly and the frog doesn't realize it's being cooked until it's too late. I think what I found is that I'm just an intellectually curious person. I enjoyed talking to my colleagues in other fields and finding out what was the cutting edge of thought in their field; what were the interesting questions they were trying to focus on. When the chance came to become president, I jumped on that.

Nowadays, most people assume that being the president of a major university is less about satisfying intellectual curiosity and more about raising money—which you've undoubtedly been good at.

In fact, more of my time and more of my staff's time is devoted to developing why we want to ask someone for money rather than actually doing the ask.

Most people who are philanthropic want to do something that has a good social return. They want a vision of how

"You have to assume that computer science will continue to grow and transform, and so, you have to be willing to learn new things."

whatever they're going to do will make a positive contribution and why it's worthy of their philanthropic attention. So we spend a lot of time thinking about how to shape a vision for the university, what we could do, why it would be distinctive, and why it would make a difference. Once you shape that vision, it's a lot easier to talk to people about how they might support it personally.

What do you make of the debate that the Knight-Hennessy program inadvertently sparked about funding institutions that are already well endowed?

I think there's two parts to it. One is ensuring people believe you'll be a good steward. Most philanthropists I know have worked really hard to acquire the ability to give to a cause, and they want to know that you're going to be a good steward of their donation. The other part is the vision. I found that it's very compelling to people when you can actually say, "You're contributing to the scholarship of a deserving student who otherwise couldn't afford to come to this institution."

I think the real challenge we're facing in the U.S. is that our public institutions are suffering under cutbacks in state funding. Philanthropy is going to have to fill in, so we'll need to articulate a message to potential donors so that they understand the need and why it's important. And then we're going to have to develop compelling opportunities, whether it's hiring faculty members in a key area or supporting students on scholarship.

Leah Hoffmann is a technology writer based in Piermont, NY.

© 2017 ACM 0001-0782/17/12 \$15.00

Distinguished Speakers Program

<http://dsp.acm.org>

Students and faculty can take advantage of ACM's Distinguished Speakers Program to invite renowned thought leaders in academia, industry and government to deliver compelling and insightful talks on the most important topics in computing and IT today. ACM covers the cost of transportation for the speaker to travel to your event.



Association for Computing Machinery

Q&A

Grooming the Leaders of Tomorrow

Former Stanford University president John Hennessy is the academic architect behind the Knight-Hennessy Scholars Program.

NO ONE WOULD accuse Stanford University's John Hennessy—co-founder of one of the first companies to commercialize RISC microprocessors, co-author of two widely used computer architecture textbooks, and the university's 10th president—of being an underachiever. Yet the man Marc Andreessen called “the godfather of Silicon Valley” stepped down from his administrative duties last year to focus on an ambitious fourth act: a multidisciplinary scholarship program aimed at grooming leaders who can solve the world's most challenging problems.

It's been just over a year since you stepped down as president of Stanford University, and it was surprisingly easy to schedule this call. I'm guessing that wouldn't have been true in 2016.

I'm certainly traveling less, and my calendar has a lot of free time by comparison. At the moment, I'm on sabbatical, which my wife says I always fail at.

For one thing, you're still deeply involved with the ambitious Knight-Hennessy Scholars Program, which aims to “build a multidisciplinary community of Stanford graduate students dedicated to finding creative solutions to the world's greatest challenges.”

When we began to think about this program, in late 2014, we saw a growing disconnect between the kind of leadership that is needed to address the really big problems—whether it's climate change or social inequality or how in-



formation technology will change the workplace—and the leadership we were getting. I think it's a widely held view that things have gotten worse around the world in the last few years, and the challenges we face are more difficult. This is true not just in government, but in the corporate and non-profit settings.

So you decided to build the program.

The first thing we did is outreach—going around the world, talking to potential future scholars about the program. Many, many students have indicated an interest, and we're hoping we can create a program that will make a real contribution toward closing this leadership void.

You also have returned to teaching.

I'm teaching a freshman seminar that's called Great Discoveries and

Inventions in Computer Science. It's an opportunity to expose students to what is intellectually deep and beautiful about computer science as a field, and to go beyond programming to everything from computability to complexity theory to cryptography to AI.

How have your experiences as president of the university shaped your views coming back to the classroom? Do you think they have given you a broader perspective?

I think so. Computing is in a phenomenally interesting place, because despite everything that's already happened with the World Wide Web and the Internet, the rise of machine learning and the use of big data are going to transform the world we live in. And that means computing is at the root of so many disciplines. In the social sciences, the use of complex, deep analysis of big data is completely changing the way we think about creating and evaluating theories about societal change and improvement. In medicine, the rise of big data provides an incredible opportunity to improve the quality of health while freeing up doctors to spend more time on the human side of helping their patients.

You have been involved with a number of interdisciplinary initiatives throughout your presidency. Let's talk about some of the intellectual and pedagogical [CONTINUED ON P. 111]

ACM Books. In-depth. Innovative. Insightful.

ACM and Morgan & Claypool Publishers present ACM Books: an all-new series of educational, research and reference works for the computing community. Inspired by the need for high-quality computer science publishing at the graduate, faculty and professional levels, ACM Books is affordable, current, and comprehensive in scope. ACM Books collections are available under an ownership model with archival rights included. We invite you to learn more about this exciting new program.

For more info please visit
<http://books.acm.org>

or contact ACM at
ACMbooks-Info@acm.org



**Association for
Computing Machinery**
2 Penn Plaza, Suite 701
New York, NY 10121-0701, USA
Phone: +1-212-626-0658
Email: acmbooks-info@acm.org



**Morgan & Claypool
Publishers**
1210 Fifth Avenue, Suite 250
San Rafael, CA 94901, USA
Phone: +1-415-462-0004
Email: info@morganclaypool.com



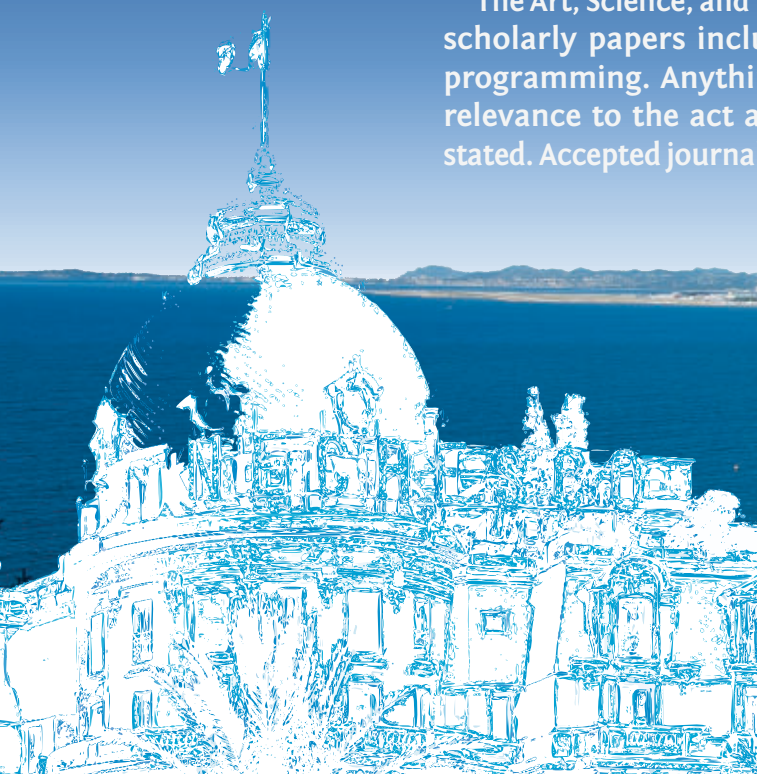
The Art, Science, and Engineering of Programming

<Programming> 2018

April 9–12, 2018 · Nice, France

The International Conference on the Art, Science, and Engineering of Programming is a new conference focused on everything to do with programming, including the experience of programming. We call it <Programming> for short. <Programming> 2018 is the second edition of the conference. Papers are welcome from any part of the programming research lifecycle, as are papers on programming practice and experience.

The Art, Science, and Engineering of Programming *Journal* accepts scholarly papers including essays that advance knowledge of programming. Anything about programming is in scope, if the relevance to the act and experience of programming is clearly stated. Accepted journal papers must be presented at the conference.



Paper Submission (issue #3)

December 1, 2017

Student Research Competition

January 1, 2018

<https://2018.programming-conference.org>



General Chair

Manuel Serrano, *Inria*

Local Organizing Chair

Tamara Rezk, *Inria*

Program Chair

Guido Salvaneschi, *TUD*

Workshops Chairs

Stefan Marr, *U. Kent*

Jennifer Sartor, *VUB*

Poster Chair

Yves Roudier, *U. Nice*

SRC Chair

Philipp Haller, *KTH*

Program Committee

Guido Salvaneschi, Davide Ancona, Alberto

Bacchelli, Shigeru Chiba, Yvonne Coady,

Susan Eisenbach, Patrick Eugster, Antonio

Filieri, Matthew Flatt, Lidia Fuentes, Richard

P. Gabriel, Jeremy Gibbons, Yossi Gil, Elisa

Gonzalez Boix, Philipp Haller, Matthew

Hammer, Felienne Hermans, Robert Hirschfeld,

Roberto Ierusalimschy, Jun Kato, Jörg Kienzle,

Neelakantan R. Krishnaswami, Ralf Lämmel,

Hidehiko Masuhara, Mira Mezini, Emerson

Murphy-Hill, Mario Südholt, Sam Tobin-

Hochstadt, Eelco Visser, Tijs van der Storm

