COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

08/2013 VOL.56 NO.08

Evolutionary Robotics

Are We Free to Code the Law?

Building Web Apps for Mobile Devices

Patient, Heal Thyself

Is Overt Censorship a Fatal Mistake?

What Is a Flagship Publication?

Association for Computing Machinery

36th International Conference on Software Engineering

ICSE 2014

Hyderabad, India • May 31 - June 7, 2014

http://2014.icse-conferences.org/

Technical Research Papers
Software Engineering Education and Training
New Faculty and Researcher Symposium
ACM Student Research Competition
New Ideas and Emerging Results
Software Engineering in Practice
Future of Software Engineering
Doctoral Symposium
Demonstrations
Workshops
Tutorials
Posters

Join us next year for ICSE 2014!
Pankaj Jalote, IIIT-Delhi, India
General Chair ICSE 2014

ACM-Infosys Foundation Award in the Computing Sciences

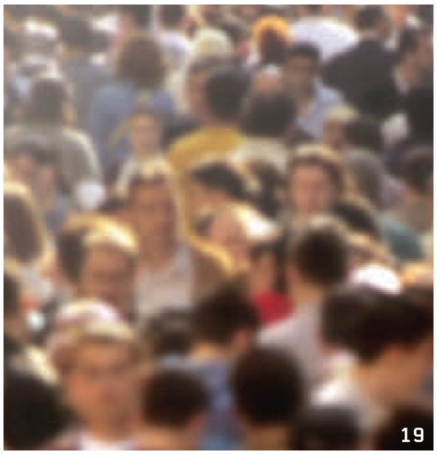
Call for Nominations

http://awards.acm.org/infosys/nominate

Association for Computing Machinery

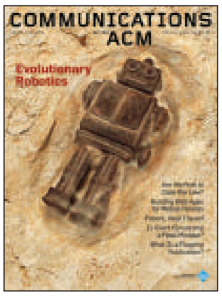
Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM



PHOTOGRAPH FROM SHUTTERSTOCK.COM

08/2012
VOL. 56 NO. 08



About the Cover: Robots resembling tin-can versions of humans have long been the stuff of entertainment: be it cinema, television, or toy-store shelves. Evolutionary robotics, however, is light-years away from that notion as it takes a biologically inspired approach to design, with few assumptions about how a machine should look and how it should behave. Cover illustration by Justin Metz.

IMAGE FROM SHUTTERSTOCK.COM; ILLUSTRATIONS BY PHIL WRIGGLESWORTH, JUSTIN METZ

Association for Computing Machinery
Advancing Computing as a Science & Profession

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.

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.

STAFF

DIRECTOR OF GROUP PUBLISHING
Scott E. Delman
publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Larry Fisher
Web Editor
David Roman
Editorial Assistant
Zarina Strakhan
Rights and Permissions
Deborah Cotton

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Directors
Mia Angelica Balaquiot
Brian Greenberg
Production Manager
Lynn D'Addesio
Director of Media Sales
Jennifer Ruzicka
Public Relations Coordinator
Virginia Gold
Publications Assistant
Emily Williams

Columnists
Alok Aggarwal; Phillip G. Armour; Martin Campbell-Kelly; Michael Cusumano; Peter J. Denning; Shane Greenstein; Mark Guzdial; Peter Harsha; Leah Hoffmann; Mari Sako; Pamela Samuelson; Gene Spafford; Cameron Wilson

CONTACT POINTS

Copyright permission
permissions@cacm.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/guidelines

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

Director of Media Sales
Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmmediasales@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

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 $100.

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 acmhelp@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

Association for Computing Machinery

Printed in the U.S.A.

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/2492007.2492008

Moshe Y. Vardi and Victor Vianu

What Is a Flagship Publication?

Communications of the ACM was launched in January 1958, with Alan J. Perlis as Editor-in-Chief. The first issue included articles such as "Tables for Automatic Computation" and "A Programmed Binary Counter for the IBM Type 650 Calculator." The new publication was declared by the Editor-in-Chief to "provide space not elsewhere available for publishing worthwhile, but possibly fragmentary, developments in the use and understanding of computers, e.g., descriptions of computer programs, computer-inspired techniques in numerical analysis, and educational efforts, to name a few." In spite of this modest initial goal, over the years, *Communications*, known also as *CACM*, became known as the "flagship magazine of the ACM." The magazine has gone through many changes over the past 55 years; see the January 2008 issue for a retrospective.

Communications was not, however, ACM's first publication. The *Journal of the ACM*, known also as *JACM*, was launched in January 1954, with Franz L. Alt as Editor-in-Chief and featured articles such as "The IBM 701 Speedcoding System," by John W. Backus. Over the years, *JACM* became known as the "flagship journal of the ACM." In July 2003, celebrating *JACM*'s 50th volume, then Editor-in-Chief Prabhakar Raghavan wrote the journal "is charged with the mission of archiving the very best research in computer science."

By 2010, however, Editor-in-Chief Victor Vianu realized that *JACM*'s aspiration of "archiving the very best research in computer science" has become very difficult to achieve. Starting with the launch of *Transactions on Mathematical Software* in 1975, ACM has developed a collection of close to 40 Transactions. While *JACM* was still a highly prestigious journal, its scope was mostly theoretical. The flourishing of the Transactions, which publish close to 1,000 articles per year, made it difficult to bridge the gap between the broad aspiration of *JACM* and its actual narrower scope. To address this gap ACM launched in the spring of 2011 a Task Force, co-chaired by us, to study and make recommendations on the appropriate mission and direction for *JACM*.

The Task Force concluded that *JACM*'s aspiration of archiving computer science's best research is no longer a realistic aspiration. ACM has many outstanding journals that are the best in their field. It is unlikely, for example, that graphics researchers will submit their best papers to *JACM* rather than to *ACM Transactions on Graphics*. Nevertheless, *JACM* can still have high aspirations. The Task Force recommended for *JACM* to "provide coverage of the most significant work on principles of computer science, broadly construed." (For details, see the revised mission statement at http://jacm.acm.org/.) The Task Force also recommended concrete steps for broadening the scope and expanding the volume of articles published in *JACM*.

The "flagship" issue was also addressed by the Task Force. A flagship is defined as the lead ship in a fleet of vessels, typically the first, largest, fastest, most heavily armed, or best known. The Task Force concluded that ACM has a sole flagship publication, *Communications*, which is, indeed, the best known and most widely distributed publication of ACM. The flagship metaphor suggests an image of a fleet led by a flag officer. In the Task Force's opinion, however, ACM's portfolio of journals and magazines (see http://dl.acm.org/) is not quite a "fleet." In fact, while ACM has an impressive collection of high-quality journals and magazines, the portfolio fails to be more than the sum of its parts. For example, one needs to simply visit the home pages of a few ACM publications; there is no indication that the individual journals are part of a cohesive portfolio. The Task Force recommended that ACM undertake to build and develop a cohesive portfolio of publications, rather than a loose collection, starting with a uniform design of journal home pages.

The flagship and fleet metaphors also suggest that it is not enough for *Communications* to be a high-quality monthly magazine. It also must be the publication that ties together ACM's publication portfolio and adds value to the portfolio as a whole. Currently, *Communications*' Research Highlights section features articles selected from (mostly) ACM research conferences, but there is no real connection between *Communications* and other ACM journals and magazines. The flagship is not leading the fleet! Discussions on how to tie the flagship better to the fleet are currently under way. We welcome your views on this matter.

Moshe Y. Vardi and Victor Vianu

Be more than a word on paper.

UNI FRESHMAN
QUALS MENTORS MAJOR
CUM LAUDE EXAMS PROFESSORS
HIGH SCHOOL GRADUATION TUMBLR
COMP SCI THESIS TECHNOLOGY GLOBAL
SEMESTER XRDS QUALITY JUNIOR
VOLUNTEER EDUCATION COLLEGE
SENIOR FINALS TECH
VOICE EDITORS CHANGE
COMMUNITY CREATIVITY CS
CHALLENGE TEAM WORK LAB ACM
LECTURE RECOGNITION SOPHMORE
DINING HALL
FACEBOOK CONTROL GRAD SCHOOLS
PRECEPT PHD SCHOLARSHIP
ACADEMIA DISSERTATION DEFENSE
LIBRARY ADVISOR TWITTER
SPRING BREAK

Join *XRDS* as a student editor and
be the voice for students worldwide.

If you are interested in volunteering as a
student editor, please contact xrds@hq.acm.org with
"Student Editor" in the subject line.

XRDS

acm Association for Computing Machinery

DOI:10.1145/2492007.2492009

Vinton G. Cerf

Computer Science Education—Revisited

ACM members span a remarkable period in the history of computing. We can point to Kelly Gotlieb, chair or co-chair of the ACM Awards committee for over 20 years, who joined ACM in 1949 when it was two years old! Next to Kelly, I feel like a newbie, having joined in 1967. And there are the most recent members, who joined us just this year, many as a result of our international initiatives.

We have a wide range of stories of the academic and professional paths that have led us to become practicing members of ACM and our profession. Similarly, our educational experiences are equally varied. Many of our members received degrees in disciplines other than computer science simply because their academic years preceded the creation of computer science departments. Indeed, Purdue University recently celebrated the 50th anniversary of its computer science department—the oldest CS department in the U.S.

At this stage of computing's evolution, it seems appropriate to reflect on what we should learn about computing and how we can or should learn it. I prefer to say "learn" rather than "teach" simply because the important metric is what one can learn about the field. Many of our most productive colleagues joined the computer science or computing field from other disciplines. Physics is often a precursor to a career in computing. Large-scale simulations, computational biology, computational linguistics, and other computational analogues along with "big data" processing occupy a good deal of the attention of today's software and hardware practitioners. Subtle and deep analyses drawing on Bayesian reasoning also occupy the thoughts of many of our colleagues.

But what about people who have no intention of entering the field? Shouldn't they have some idea of how programming works? How operating systems work? How networks and their layered protocols work?

Among the initiatives that ACM has been pursuing is to make computer science acceptable as a core science along with mathematics, physics, biology, and chemistry. This is especially critical in secondary schools where, with few exceptions, computing classes tend to be optional and not a substitute for any other discipline. In many advanced programs, it is a requirement to have a certain number of credits in science, for example. It is ACM's position that computer science should have equal standing. Moreover, the curriculum should include some serious exposure, inter alia, to programming, systems, languages, and computer architecture. The idea is not necessarily to turn students into professional computer engineers and scientists, but to expose them to the richness of computer science and to help them appreciate the potential nascent in computers and programmable systems.

In many advanced programs, it is a requirement to have a certain number of credits in science. It is ACM's position that computer science should have equal standing.

A detailed report on this topic can be found at http://csta.acm.org/Advocacy_Outreach/sub/Inroads_Stephenson-Wilson.pdf. I strongly urge you to read this report and then to explore ACM's website page on education: http://www.acm.org/education.

Reforming K–12 education to incorporate serious computer science seems vital to producing an informed public that has a deeper appreciation for the power of computing than video games and social networking. There are, no doubt, countless opportunities for computing professionals to engage in this effort, by lending their support and time to the effort to reform K–12 curriculum content and to make visible to young people the excitement of discovering what computing can accomplish. The discipline of writing and debugging software, of creating simulations or interactive applications has the potential to draw many into the profession, or at least to provide even more with a sense of the core role computing is playing and will play in the decades ahead.

As the Internet of Things becomes reality and software appears in every appliance, building, and vehicle, we have a societal interest in promoting understanding of and interest in our discipline.

Vinton G. Cerf, ACM PRESIDENT

ACM Awards

Call for Nominations

http://awards.acm.org/nominate

Association for
Computing Machinery

Advancing Computing as a Science & Profession

DOI:10.1145/2492007.2492011

Is Computing Science?

PETER J. DENNING'S Viewpoint "The Science in Computer Science" (May 2013) explored the ongoing dispute over scientific boundaries within computer science. The root word in Latin for science is "knowledge," and computer science likewise concerns knowledge. However, the boundaries separating the sciences, and knowledge in general, have never been clear and definite.

In the mid-20th century, John von Neumann was emblematic of the idea that there are no clear boundaries. "Mathematician" is the word most often used to describe him, though he was also a physicist, economist, engineer, game theorist, and meteorologist, as well as computer scientist, even though computer science did not exist as a discipline at the time.

The term "von Neumann architecture" reflects how von Neumann's professional life defined the principles of modern digital computing. Was he a computer scientist? If we could ask him, he would say yes, because he appreciated that he used computing as a tool, even though such an assertion would have alienated many colleagues at the Institute for Advanced Study in Princeton, NJ. He ignored the historical boundaries of the disciplines, but his contributions expanded them all because knowledge imposes no restrictions on what or how knowledge is applied. In this light, the tool makes the man. Can one be a surgeon without being able to use a scalpel, an astronomer without being able to use a telescope, or a microbiologist without being able to use a microscope?

The reason computing is so exciting today is precisely because such boundaries are irrelevant. Before Google, who would have imagined a "search engine" would become a multibillion-dollar industry or that computing power combined with powerful telescopes would explore for Earth-like planets light-years away? The power of computing is itself the power of knowledge.

If there were indeed clear boundaries within the sciences, Thomas S. Kuhn's 1962 book *The Structure of Scientific Revolutions* exposed them as untenable. His study of what constitutes "normal" vs. "revolutionary" science has been controversial ever since because drawing boundaries is nearly impossible.

Computing practitioners who feel slighted when someone says their profession is less than scientific should calm themselves. Computing is at the heart of the expansion of knowledge in practically every discipline, without regard to prior boundaries. Unlike any other tool ever devised, computing manages to straddle Boolean logic, materials science, control of electron flow, manufacturing knowhow, and semanticity. Moreover, it has no inherent size, with Moore's Law applying regardless of scale. Semanticity means computers are the first machines to be able to store and manipulate symbols that are also meaningful to humans.

Knowledge is at the heart of computing, and knowledge has but one boundary, between itself and ignorance and superstition. Von Neumann made no effort to justify his professional pursuits, recognizing that knowledge is but one thing, available to all who think.

Francis Hsu, Rockville, MD

Author's Response:

Hsu eloquently argues on behalf of my main conclusion—that computing science cuts through many fields while enriching them all with an understanding of information and information transformations—a conclusion that will eventually be widely accepted. The challenge in the near term is that many K–12 school systems do not recognize computing as a science, nor do they have computing courses, something many people are working to change. I hope our Ubiquity symposium (http://ubiquity.acm.org) provides them some needed ammunition.

Peter J. Denning, Monterey, CA

Reconciling ACM Bibliometric Numbers

Scott E. Delman's Publisher's Corner column "A Few Good Reasons to Publish in *Communications*" (May 2013) included an unexplained oddity in its otherwise interesting bibliometric numbers. The figure said *Communications* has published 11,257 articles, of which 11,256 are available for download. Is there really exactly only one article not available for download? And if so, which one?

Mark J. Nelson, Copenhagen, Denmark

Editor's Response

Upon investigation, the ACM Digital Library team discovered we were indeed shy one .pdf document. A 200-word announcement listed in the Table of Contents of the April 2007 Communications *lacked an accompanying .pdf—hence the discrepancy between publication count and download count. We have rectified the omission and thank you for your careful reading and for bringing it to our attention.*

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.

DOI:10.1145/2492007.2492012 http://cacm.acm.org/blogs/blog-cacm

Teaching Programming the Way It Works Outside the Classroom

Philip Guo offers programmers 'Opportunistic Programming' tips that typically are not shared in school.

Philip Guo
Teaching Real-World Programming
http://cacm.acm.org/blogs/blog-cacm/159263-teaching-real-world-programming/fulltext
January 7, 2013

In this post, I describe a ubiquitous style of programming that, to my knowledge, has never been formally taught in the classroom.

In most programming classes, students write programs in a single language (e.g., Java, Python) and its standard library; they might use a well-documented third-party library for, say, graphics. Students fill in skeleton code templates provided by instructors or, at most, write a few chunks of code "from scratch." Specifications and interfaces are clearly defined, and assignments are graded using automated test suites to verify conformance to specs.

What I just described is necessary for introducing beginners to basic programming and software engineering concepts. But it bears little resemblance to the sorts of programming that these students must later do in the real world.

Over my past decade of programming, I have built research prototypes, extended open-source software projects, shipped products at startups, and engaged in formal software engineering practices at large companies. Regardless of setting, here are the typical steps my colleagues and I take when starting a new project:

1. Forage: Find existing snippets of code to build my project upon. This might include code I wrote in the past or that colleagues sent to me in various stages of bit-rot. If I am lucky, I can find a software library that does some of what I want; if I am really lucky, then it will come with helpful documentation. Almost nobody starts coding a real-world project "from scratch" anymore; modern programmers usually scavenge parts from existing projects.

2. Tinker: Play with these pieces of existing code to assess their capabilities and limitations. This process involves compiling and running the code on various inputs, inserting "print" statements to get a feel for when certain lines execute and with what values, and then tweaking the code to see how its behavior changes and when it breaks. (Now loop between steps 1 and 2 until I am satisfied with my choice of building blocks for my project. Then move on to step 3.)

3. Weld: Try to attach ("weld") pieces of existing code to one another. I might spend a lot of time getting the pieces compiled and linked together due to missing or conflicting dependencies. Impedance mismatches are inevitable: Chances are, the code pieces I have just welded together were never designed to "play nicely" with one another, or to suit the particular needs of my project.

4. Grow: Hack up some hard-coded examples of my new code interfacing with existing "welded" code. At this point, my newborn code is sloppy and not at all abstracted, but that is okay—I just want to get things working as quickly as possible. In the process, I debug lots of idiosyncratic interactions at the seams between my code and external code. Wrestling with corner cases becomes part of my daily routine.

5. Doubt: When implementing a new feature, I often ask myself, "Do I need to code this part up all by myself, or is there some idiomatic way to accomplish my goal using the existing code base or libraries?" I do not want to reinvent the wheel, but it can be hard to figure out whether existing code can be molded to do what I want. If I am lucky, I can ask the external code's authors for help; but I try not to get my hopes up because they probably did not design their code with my specific use case in mind. The gulf of execution is often vast: conceptually simple features take longer than expected to implement.

6. Refactor: Notice patterns and redundancies in my code and then create abstractions to generalize, clean up, and modularize it. As I gradually refactor, the interfaces between my code and external code start to feel cleaner, and I also develop better intuitions for where to next abstract. Eventually I end up "sanding down" most of the rough edges between the code snippets that I started with in step 4.

(Now, repeat steps 4 through 6 until my project is completed.)

I do not have a good name for this style of programming, so I would appreciate any suggestions. The closest is *Opportunistic Programming*, a term my colleagues and I used in our CHI 2009 paper where we studied the information foraging habits of web programmers. Also, I coined the term *Research Programming* in my Ph.D. dissertation, but the aforementioned six-step process is widespread outside of research labs as well. (A reader suggested the term *bricolage*.)

Students currently pick up these hands-on programming skills not in formal CS courses, but rather through research projects, summer internships, and hobby hacking.

One argument is that the status quo is adequate: CS curricula should focus on teaching theory, algorithm design, problem decomposition, and engineering methodologies. After all, "*CS != Programming*," right?

But a counterargument is that instructors should directly address how real-world programming—the most direct applications of CS—is often a messy and ad hoc endeavor; modern-day programming is more of a craft and empirical science rather than a collection of mathematically beautiful formalisms.

How might instructors accomplish this goal? Perhaps via project-based curricula, peer tutoring, pair programming, one-on-one mentorship, or pedagogical code reviews. A starting point is to think about how to teach more general intellectual concepts *in situ* as students encounter specific portions of the six-step process described in this post. For example, what can "code welding" teach students about API design? What can refactoring teach students about modularity and testing? What can debugging teach students about the scientific method?

My previous CACM post, "Teaching Programming To A Highly Motivated Beginner," describes one attempt at this style of hands-on instruction. However, it is still unclear how to scale up this one-off experience to a classroom (or department) full of students. The main challenge is striking a delicate balance between exposing students to the nitty-gritty of real-world programming, while also teaching them powerful and generalizable CS principles along the way.

Please post your thoughts as comments or email me at philip@pgbovine.net.

Readers' comments:

Over the last year, I've spent a fair number of cycles thinking about the disconnect between how people actually build code and how we teach programming in classrooms. This article hits on some of the same points I've thought about, especially with regard to the fact that programming today is increasingly about information foraging and composition of existing solutions. The process is defined by experimentation and iteration.

It's also more of a social task, with online resources providing and peers providing increasing amounts of support. I've been thinking about these factors as I start my new job, slinging code for the Googs. As with many large companies there is a ton of existing code, and I have set up some pair programming sessions with folks on my team. These folks have a ton more experience with the tools that I will be using than I do. They can point me to resources I would have a hard time finding myself.

I wonder if coursework can mirror reality here. One idea that might work is the creation of a new course that paired undergrads across years, perhaps sophomores and seniors. The seniors should have more experience working on a project through coursework and internships.

One option is to provide open-ended projects based on the skills the seniors bring to the table. A senior of app programming experience might be asked to create a new Android or iPhone game, whereas one that had been working on building systems might be asked to create a large-scale data processing app on top of Hadoop. Another option might be to have a two-stage course focused on a specific project, separated between years. Current sophomores would have to come back in two years and share their knowledge.

Of course this is one of many solutions, but I think we can do a better job of teaching the social aspects of building software.

—Kayur Patel

I think it is difficult to mirror real-world programming scenarios in the classroom, because the classroom isn't the real world. But that doesn't mean that a programming course for beginners has to be structured along short, isolated programming exercises using a single programming language. For a few years now we have been pursuing a first-year programming course under the headline "object-oriented development of web applications." Of course, during this class the students program mainly using a single language, in this case it is Smalltalk. But because a web application is being developed they have to learn the usage of a web framework (Seaside), HTML, and CSS from the beginning. Later on, JavaScript supervenes.

During the whole course, each student develops one single web application. The development is guided by a series of online lectures. For the first steps unit tests are provided by the instructors, but during further progress of the project the students have to develop their own unit tests.

At the beginning of the second semester, the students have to build teams with three members. They learn to develop software cooperatively and to merge components developed independently.

As the complete course content is presented by online lectures, the instructors can play a role as coaches instead of lecturers. This allows them to adjust heterogeneous previous knowledge of the students.

Of course, not all real-world programming issues can be anticipated this way, but we think this kind of programming course allows many more typical problems to be covered than during a conventional one.

—Johannes Brauer

Philip Guo is a visiting research scientist at edX. In the fall, he will join the Massachusetts Institute of Technology (MIT) Computer Science and Artificial Intelligence Laboratory (CSAIL) as a postdoctoral scholar.

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD13

You can join ACM in several easy ways:

Online	Phone	Fax
http://www.acm.org/join	*+1-800-342-6626 (US & Canada)* *+1-212-626-0500 (Global)*	*+1-212-944-1318*

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:
http://www.acm.org/membership/L2-3/

Special rates for members of sister societies:
http://www.acm.org/membership/dues.html

Please print clearly

Name

Address

City	State/Province	Postal code/Zip
Country	E-mail address	
Area code & Daytime phone	Fax	Member number, if applicable

Purposes of ACM

ACM is dedicated to:
1) advancing the art, science, engineering, and application of information technology
2) fostering the open interchange of information to serve both professionals and the public
3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/about/code-of-ethics

choose one membership option:

PROFESSIONAL MEMBERSHIP:

o **ACM Professional Membership: $99 USD**

o **ACM Professional Membership plus the ACM Digital Library: $198 USD ($99 dues + $99 DL)**

o **ACM Digital Library: $99 USD (must be an ACM member)**

STUDENT MEMBERSHIP:

o **ACM Student Membership: $19 USD**

o **ACM Student Membership plus the ACM Digital Library: $42 USD**

o **ACM Student Membership PLUS Print *CACM* Magazine: $42 USD**

o **ACM Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 USD**

All new professional members will receive an ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include $40 toward a subscription to *Communications of the ACM*. Student membership dues include $15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

o Visa/MasterCard   o American Express   o Check/money order

o Professional Member Dues ($99 or $198)	$ ________
o ACM Digital Library ($99)	$ ________
o Student Member Dues ($19, $42, or $62)	$ ________
Total Amount Due	$ ________

Card #   Expiration date

Signature

Science | DOI:10.1145/2492007.2492013

Samuel Greengard

A New Approach to Information Storage

Disk drives and solid-state drives have long served as the foundation for computer storage, but breakthroughs in molecular and DNA science could revolutionize the field.

WHEN GEORGE CHURCH, a geneticist at Harvard Medical School, decided to produce 70 billion copies of his book, *Regenesis: How Synthetic Biology Will Reinvent Nature and Ourselves*, he skipped printing presses, Kindles, and hard drives. The professor of genetics instead turned to a most unlikely medium: DNA, the same long molecule that serves as the building block for life on Earth. "It has worked remarkably well as a storage medium for 3.5 billion years," he says.

In Church's case, a team of researchers used sequencing technology to format his 54,000-word book (with words, images, and a JavaScript program, it came down to 5.27 megabits, or 658.75 bytes) at a density of 5.5 petabytes per cubic millimeter. While the physical volume of 70 billion physical copies of his book would fill nearly 3,500 New York City Public Libraries (including all branches), and a digital version would require somewhere in the neighborhood of 46 storage devices with 1TB drives, all those copies of Church's book fit on a piece of DNA no larger than a speck of dust. What's more, the copies will last hundreds of thousands of years—perhaps even a million years—and do not require any special handling or temperature conditions.

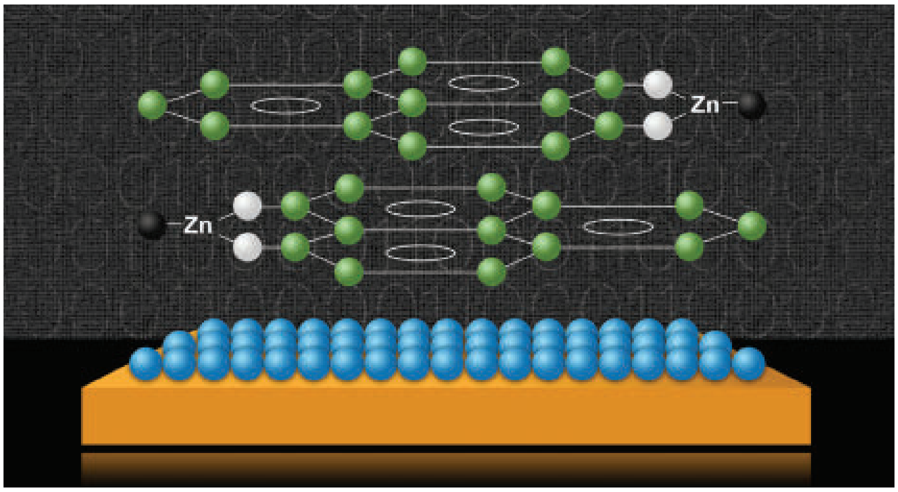
Welcome to the emerging world of data storage. While hard drive and solid-state drive manufacturers are attempting to increase storage densities and push the limits on speed and performance, a handful of researchers around the world are hard at work on the next generation of systems and devices that would crash standard thinking about storage. Some, like Church and the European Bioinformatics

Harvard geneticist George Church shows the amount of space needed to store 20 million copies of his book within DNA.

Institute (EBI), are focusing on DNA. Others, including a research group at the Massachusetts Institute of Technology (MIT), are examining molecular storage methods.

Both approaches have begun to take shape over the last few years—although the feasibility of DNA storage was first demonstrated in 1988. Over the next decade, new approaches to data storage could transform the way organizations, and society, manage and store huge volumes of data.

For perspective, all the data humans produce in a year could fit into about four grams of DNA. "There is an opportunity to create storage systems that are a million to a billion times more compact than existing technology and provide a level of longevity that is unheard of today," Church points out.



An artist's depiction of graphene fragments, flat sheets of carbon attached to zinc atoms, which may be used in the manufacture of molecular memories.

The DNA of Storage

The need for more efficient data storage methods is rooted in today's radically changing world. According to IBM, humans collectively produce about 2.5 exabytes of data each day; market research firm IDC says roughly three zettabytes of data exist in the digital world. Remarkably, 90% of the data in the world has been created over the last two years alone, say researchers at IBM. All this data requires increasingly large data centers and storage networks. It also presents challenges as storage devices and media change and data technologies become obsolete and prone to failure.

Researchers hope to significantly alter the equation. Church and fellow researchers, including Sri Kosuri, a senior scientist at the Wyss Institute, and Yuan Gao, an associate professor of biomedical engineering at Johns Hopkins University, are forging into new territory with DNA storage research. They used sophisticated sequencing techniques to encode Church's book in 96-bit blocks, each containing a 19-bit address to assist with the reassembly process.

The data was built from code based on the four constituents of DNA: adenine (A), guanine (G), cytosine (C), and thymine (T), and converted to binary code. The non-living DNA contained 54,898 data blocks—each stored on an individual strand of protein. The team then sent the data to Agilent Technologies, which used a 3D printer to attach the data to the DNA strands and build a physical storage device. Then the team accurately decoded the text and read it back. Remarkably, a billion copies of the book easily fit into the moisture on the bottom of a glass or small tube.

Church says the DNA storage method is ideal for archival copies of huge datasets. The challenge is speeding the DNA fabrication process—possibly by turning to optical technology that writes with light enzymatically. For now, the writing process is expensive and slow, but the situation could flip as researchers invent better writing technologies. "We could see million-fold improvements in writing technology," he says. "Unlike Moore's Law, which results in improvements at a factor of about 1.5 per year, DNA sequencing is advancing at a 10-fold increase per year. This could translate into commercially viable technology within five years."

These writing and reading systems could attach to a computer using a USB or similar port. Although this storage technology probably would not be practical for everyday use—at least in the foreseeable future—it creates a media format that can last over the long term while eliminating the need to change media every few years as new devices and technologies supplant previous generations (such as when tape transitioned to CDs, then DVDs, and later to digital file formats). "This eliminates backward compatibility issues related to new generations of technology," Church explains. What's more, "It is possible to store the data for half a million years without electricity." Indeed, the technology could, in some cases, eliminate vast storage networks and provide significant environmental benefits.

Another group examining DNA storage is the European Bioinformatics Institute (EBI) in Hinxton, England. In January 2013, scientists there reported they had encoded DNA with a 26-second audio clip of Martin Luther King's "I Have a Dream" speech, a photograph, an academic paper, and 154 sonnets from Shakespeare. The DNA was dried onto glass sheets or vials. Researchers were able to retrieve the data with a 99.99% accuracy rate. Since then, they have corrected a "biological glitch" and they can now achieve near-100% accuracy.

In addition to bringing down the current cost of writing data to DNA—about €12,500 ($16,365) per megabyte—there is the challenge of building a system or technology that manages the data over long time spans. "One of the keys to making DNA storage work is establishing appropriate metadata and indexing systems," says Nick Goldman, group leader for EBI. "We need a Rosetta Stone equivalent that can span hundreds or thousands of years and make sure all the data is directed to the right file, device, or system as it is needed."

Other scientists also are examining possibilities related to DNA storage. For example, a research group at Stanford University has experimented

GRAPHIC BY CHRISTINE DANILOFF/MIT

with using living DNA cells in *E. coli* bacteria to store digital code. This approach could aid in studying cancer, aging, and organismal development, the group reports, although the approach is not particularly efficient or desirable for holding massive volumes of data. Church notes that if the living cells do not find an evolutionary advantage to the data, they will begin mutating it, and at some point, they will destroy it.

Molecular Data Storage

Next-generation storage techniques are advancing in other ways. At MIT, a group of researchers is diving into the realm of molecular storage. The group has found a way to create a new type of supramolecule from molecules specially assembled by the Indian Institute of Science Education and Research in Kolkata. This supramolecule binds two different types of atoms: fragments of graphene, comprised of thin sheets of carbon atoms, with zinc atoms. When these atoms are placed on a magnetic surface, the resulting magnetized supramolecule is about one nanometer in size and able to store data at a density of 1,000 terabytes per square inch (compared to a maximum capacity of less than one terabyte of data per square inch in current hard drive technology).

The experimental technology works in a somewhat different way than standard magnetic drives. Researchers placed a thin film of the molecular material they developed on a ferromagnetic electrode, and added a second ferromagnetic electrode on top. When a relative change in one electrode's magnetic orientation occurs, there is a sudden increase or decrease in the system's conductivity. These two states represent the 1s and 0s of binary code.

However, the MIT researchers observed two jumps in conductivity—even when the supramolecule had only one associated ferromagnetic electrode, rather than the pair. "This occurrence came as a complete surprise," says Jagadeesh S. Moodera, a senior research scientist in the MIT Department of Physics. The ability to alter the conductivity of the molecules with only one ferromagnetic electrode could drastically simplify the manufacture of molecular memory.

"The problem with today's physical storage devices is that we are approaching their physical limits."

A 1,000-times increase in storage density could redefine everything from data centers to personal devices. "The problem with today's physical storage devices is that we are approaching their physical limits," says Karthik V. Raman, a research scientist at IBM India and part of the MIT team that invented the molecular storage technology. "Molecular storage could offer far better performance in terms of data retention, densities, and power use. It could result in much more powerful and smaller devices. A device the size of an iPhone could have a staggering amount of storage capacity."

Moodera says there is still considerable work to be done on the concept. While scientists have demonstrated the technology works, they eventually hope to show two stable and nonvolatile states for the molecules. In addition, the technology currently operates at a temperature of -9 degrees Fahrenheit—so-called "room temperature" in physics. Researchers will have to find way to build the storage structure at higher temperatures to make it commercially viable.

However, the challenges do not end there. The researchers must also find a way to boost conductivity differences from the current 20% range to perhaps 50% or more. Getting to this point could take a decade or more, and will require both material innovation and fabrication advances. "We need to investigate further so we can achieve a deeper understanding at the molecular level," Moodera explains.

Storage: The Next Generation

In the end, it is not so much a question of *if* next-generation storage technologies will go mainstream, but *when*. As we continue to amass increasing stores of data, the need for new storage technologies becomes increasingly clear. Molecular and DNA storage could become the vehicles of choice, or something new could appear. Either way, "New generations of vastly more efficient storage systems could fundamentally change the way we approach data, manage complex tasks, and approach computing," Raman explains.

For now, researchers are looking to fill in the gaps in order to produce commercially viable systems. They are tapping expertise in every discipline from biology and quantum physics to software development to assemble all the pieces and build the storage medium of the future. Says MIT's Moodera: "The goal is to explore different molecules, different configurations, and different ways of applying computing technology. Although these future storage mediums are remarkably complex, we are on the doorstep of developing remarkable systems that will redefine the way we manage, store, and use data." C

Further Reading

Raman, K.V., Kamerbeek, A.M., Mukerjee, A, Atodiresel, N., Sen, T.K., Lazić, P., Cacluc, V., Michel, R., Stalke, D., Mandal, S.K., Blügel, S., Münzenberg, M., Moodera, J.S.,
Interface-engineered templates for molecular spin memory devices, Nature, 493, 509-513, Janaury 2013. http://www.nature.com/nature/journal/v493/n7433/full/nature11719.html

Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E.M., Sipos, B., Birney, E.,
Towards practical, high-capacity, low-maintenance information storage in synthesized DNA, Nature, January 2013. http://www.nature.com/nature/journal/vaop/ncurrent/full/nature11875.html

O'Driscoll, A., Sleator, R.D.,
Synthetic DNA: The Next Generation of Big Data Storage, Bioengineered, May/June 2013. http://www.landesbioscience.com/journals/bioe/2013BIOE-NV-43.pdf.

Church, G.M., Gao, Y., Kosuri, S.,
Next-Generation Digital Information Storage in DNA., Science, Sept. 2012, Vol. 337, no. 6102, P. 1628. http://www.sciencemag.org/content/337/6102/1628.abstract

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

Technology | DOI:10.1145/2492007.2492014 Alex Wright

Patient, Heal Thyself

New handheld medical diagnostic tools promise more efficient, lower-cost healthcare—but at what price?

When Walter de Brouwer's five-year-old son suffered a brain injury after falling from a 36-foot-high window in 2005, the Belgian inventor and his wife spent 10 long weeks in the intensive care unit, waiting helplessly while the doctors and nurses pored over a bewildering array of medical data.

"It was frustrating to feel so in the dark," de Brouwer recalls. As the days dragged on, he spent more and more time poring over the data, slowly mastering the arcana of vital signs and medical record-keeping as he tried to make sense of his son's condition. Along the way, he began cultivating a new product idea: an easy-to-use handheld device that would allow patients to gather and interpret their own vital signs.

In 2011, de Brouwer launched Scanadu, a software company whose flagship product, Scout, captures five critical vital signs via a small sensing device about the size of a mouse, then transmits that data via Bluetooth to a smartphone equipped with a special diagnostic app. Backed by technology luminaries like Stephen Wolfram and Nicholas Negroponte, the company seems well-poised to bring medical data-gathering to the masses within the next few years.

Scanadu is scarcely alone. In recent years, a slew of new companies have emerged with new handheld medical data-gathering devices: from the popular Fitbit to smartphone-enabled ultrasound machines, portable glucose monitors, and even handheld electrocardiogram devices.

In a widely read column in *TechCrunch* last year, former Sun Microsystems CEO and influential venture capitalist Vinod Khosla predicted the emergence of what he called "Dr. Algorithm," a system capable of gathering patient data from an array of handheld devices and querying a vast storehouse of connected patient data and medical literature to diagnose common medical conditions. "Eventually, we won't need the average doctor and will have much better and cheaper care for 90 to 99 percent of our medical needs," he wrote.

While the technological pieces seem to be falling into place, the path to cheaper medical care is fraught with obstacles. In addition to the technical challenges of writing diagnostic software and integrating it with existing medical records systems, developers must also come to terms with a host of thorny ethical and regulatory issues—not to mention a looming backlash from doctors who are just beginning to understand the implications of all this DIY medical data gathering.

In spite of these challenges, new companies are flocking to the fast-growing market for wireless healthcare and services, currently estimated at $9.6 billion. The pace of development received a further boost in January 2012 when the X PRIZE Foundation announced a new $10-million award, sponsored by Qualcomm, for the creation of a "medical tricorder"—a Dr. McCoy-worthy device capable of diagnosing 15 common diseases like diabetes, pneumonia, and tuberculosis.

In January 2012, the X Prize Foundation announced a new $10-million award for the creation of a "medical tricorder."

The prize's senior director, Mark Winter, feels the time is right to bring Star Trek-level diagnostic tools to the market. "We're seeing a remarkable convergence of different technologies to solve big, complex problems," he says. Specifically, he points to the growing availability of sophisticated wireless sensors, along with the rapid acceleration of smartphone processors, as the key technological underpinnings that could make the medical tricorder a reality.

Dr. Scott Jansen recently worked on one of the X PRIZE teams, building on his earlier work in creating an open source "science tricorder." Although he is no longer actively involved with the medical team, he feels confident that the time is ripe for such a device.

"Five years ago, it was difficult to find sensors that would fit into a handheld device," he says, and those that he could find often drained power resources and gave off poor signals.

Today, that situation has changed dramatically. Sensors are getting smaller—meaning more of them can fit in a small device—and, more importantly, they are getting cheaper. Modern sensors are power efficient as well, often featuring onboard analog-to-digital converters to boost their signal quality.

The availability of high-quality off-the-shelf sensors—like spectrometers or high-energy particle detectors—has dramatically changed the engineering equation.

"While integrating a broad array of sensing modalities into a single device can be challenging, having manufacturers develop single-chip smart sensing products really reduces months of engineering work down to a few hours of integration," says Dr. Jansen.

A Redmond, WA-based start-up

Scanadu's Scout device captures vital signs that are transmitted to a smartphone equipped with a diagnostic app.

called MobiSante is taking advantage of increasing smartphone processor speeds and outsourced sensor manufacturing to develop a handheld ultrasound scanning solution that sells for less than one-tenth the cost of traditional ultrasound systems. By using a scanning device tethered to a smartphone and powered via a USB connection, the system leverages readily available technologies to drive down the cost of ultrasound scans.

"We're riding Moore's Law," says MobiSante CTO David Zar, who notes that improvements in processor speeds have enabled the team to produce ultrasound images on a smartphone that would have been considered state of the art just 10 years ago.

"We want to bring the testing to the patient," says Zar, who spent the better part of two decades at Washington University in St. Louis perfecting what he calls "dirt-cheap ultrasound."

Although the MobiSante system cannot produce the high-resolution 4D scans of top-of-the-line ultrasound machines, it nonetheless produces a level of fidelity that can be used by primary care physicians as an early screening tool. The system promises to put ultrasound technology into the hands of doctors who might not otherwise have had the resources to invest in expensive ultrasound machines—thus making ultrasound scans available to a much wider population of patients.

As the growing availability of cheap sensors and faster processors reduces the economic barriers to entry, companies entering the market will increasingly need to focus on differentiating their products at the software layer.

"Our secret sauce in in our algorithms," says de Brouwer. "Putting all the sensors in the right place and creating algorithms to ensure accuracy is very precise and challenging. It's not only electrical engineering and mechanical engineering, but also imaging, physics, and molecular diagnostics."

Zar agrees that software will become increasingly important in a marketplace of products built on increasingly commoditized hardware. Beyond the usual challenges of application development and systems integration, he also points to the heightened importance of quality assurance in the highly regulated world of medical devices.

"We have a higher bar," says Zar. While the United States Food and Drug Administration (FDA) does not review anyone's software code, the agency does require that makers of certified medical devices demonstrate their commitment to following a quality management program. "You have to keep meticulous records," he says.

Medical device makers also must often navigate a thicket of regulatory approvals necessary to bring their products to market. "The problem is that there are too many devices on the market that claim medical repercussion of actions, but lack accuracy," says de Brouwer, whose firm is now working closely with the FDA to certify its devices.

Just as self-service legal software has created new quandaries for lawyers, consumer medical applications run the risk of resistance from the medical community, which has traditionally resisted efforts to empower patients with their own medical data.

As patients gain more and more access to these tools, many of them may feel tempted to play doctor for themselves—especially given a looming shortage of qualified doctors and the rising costs of healthcare. Yet the days of diagnosing an ailment with the swipe of a touchscreen remain a long way off.

While the Internet has long since opened the floodgates to medical literature (who among us has not given in to the occasional bout of Google-powered hypochondria?), doctors and hospitals have so far kept a tight grip on personal medical records.

Diagnosing illness is a serious business, after all—and doctors will likely cast a wary eye at some of these devices, as well they probably should. The road to self-service medicine is fraught with perils. Putting diagnostic technology in the hands of non-specialists could introduce new, potentially life-threatening risks for some people.

Doctors have long enjoyed a so-called asymmetric information advantage: a privileged position predicated on their specialized knowledge and a tightly held grip on personal medical data. But the balance of power seems to be shifting—in part due to the emergence of these devices, as well as the growing "quantified self" movement that has spurred a growing demand for personal health monitoring products.

"This is about the democratization of medicine," says Dr. Eric Topol, a cardiologist and geneticist at Scripps Health and author of *The Creative Destruction of Medicine*. "We are moving towards the individual taking charge."

On a recent flight, Topol used his AliveCor portable electrocardiogram device to determine that a fellow passenger was having a heart attack. He told the pilot to land the plane as soon as possible; that quick diagnosis likely saved the passenger's life. However, Topol is a trained cardiologist; what happens when ordinary citizens start trying to use their smartphones to make their own diagnoses?

"The medical community is quite threatened by this," says Topol. Traditionally resistant to change, and highly invested in preserving its own authority—not to mention its billings—these tools pose a potentially major disruption to the entire medical economy.

Topol also shares his fellow doctors' concerns about the potential danger of patients trying to interpret medical data for themselves. "There are big risks," says Topol. "You have to validate that this is a good thing. You don't want to have all this unplugged medicine completely unbridled."

For this reason, the FDA exerts tight control over the transmission of medical data. While data-gathering devices like thermometers, scales, and blood pressure cuffs are not regulated, FDA regulations kick in the moment a device attempts to deliver a diagnosis, or transmits data to a medical provider.

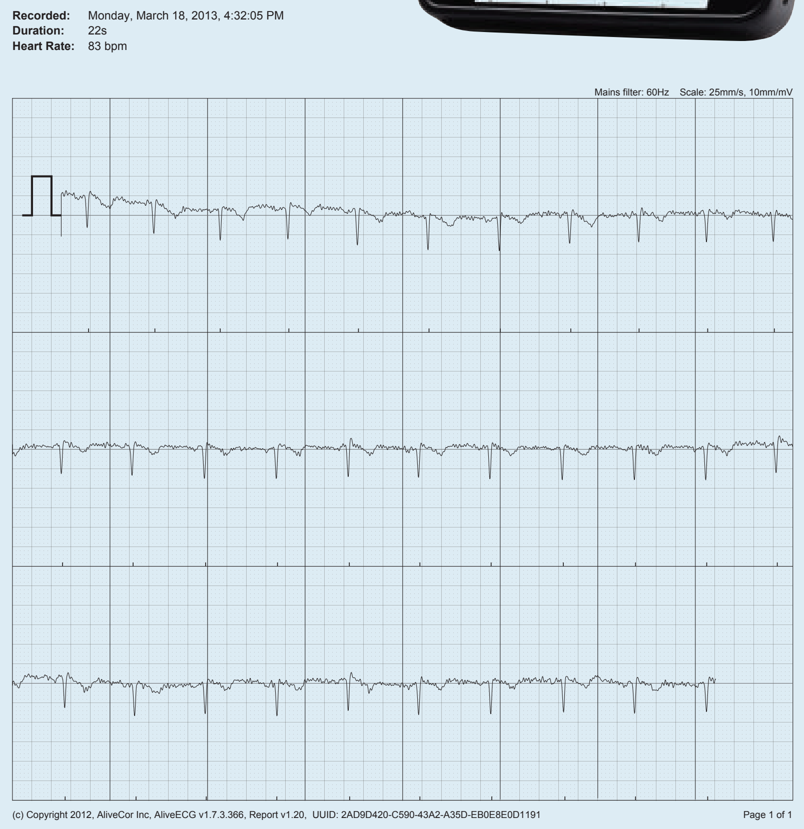
As new technologies emerge, the FDA is trying to move quickly to make sure that its regulatory bodies adapt to technological change. As is so often the case, however, technology seems to be running a few steps ahead of the regulatory apparatus.

Given the looming shortage of doctors in many countries, more distributed data-gathering technologies may ultimately transform the role of physicians into a kind of über-QA function, responsible for ensuring the quality of care and analyzing data gathered at a distance, while doing less and less hands-on healing.

It may be too soon to predict the demise of the traditional check-up, but there seems to be little question that the emergence of handheld medical technologies increasingly will transform the dissemination of medical information and, over time, shift more and more data into the hands of patients.

As these issues begin to shake out, it seems likely that self-service medical technologies will continue to challenge the healthcare status quo. Says de Brouwer: "There is so much we don't know about our bodies outside of hospital walls." C

The AliveCor handheld heart monitor, right, and a sample ECG (below) that can be emailed to doctors and patients.


Recorded: Monday, March 18, 2013, 4:32:05 PM
Duration: 22s
Heart Rate: 83 bpm

Mains filter: 60Hz Scale: 25mm/s, 10mm/mV

(c) Copyright 2012, AliveCor Inc, AliveECG v1.7.3.366, Report v1.20, UUID: 2AD9D420-C590-43A2-A35D-EB0E8E0D1191 Page 1 of 1

Further Reading

Khosla, Vinod.
"Do We Need Doctors or Algorithms," *TechCrunch* (January 10, 2012), http://techcrunch.com/2012/01/10/doctors-or-algorithms/

2011. *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare.* ACM, New York, NY, USA.

Predrag Klasnja and Wanda Pratt.
2012. Methodological Review: Healthcare in the pocket: Mapping the space of mobile-phone health interventions. *J. of Biomedical Informatics* 45, 1 (February 2012), 184-198. DOI=10.1016/j.jbi.2011.08.017 http://dx.doi.org/10.1016/j.jbi.2011.08.017

"Ultrasound Imaging More Portable, Affordable with USB-Based Probes," Microsoft Research White Paper, http://research.microsoft.com/en-us/collaboration/focus/health/msr_ultrasound.pdf

Alex Wright is a writer and information architect based in Brooklyn, NY.

Society | DOI:10.1145/2492007.2492015 Paul Hyman

Software Aims to Ensure Fairness in Crowdsourcing Projects

The debate rages on about whether crowdsourcing is a win-win for workers, as well as for employers.

ARE WORKERS WHO participate in the highly distributed microlabor online system known as crowdsourcing treated fairly? And what about the crowdsourcing employers?

It is not a new topic of debate, but a new paper presented at the ACM SIGCHI Conference on Human Factors in Computing Systems in Paris in April is likely to heat up the discussion considerably, especially since one of the paper's two authors urges computer professionals to take a harder look at crowdsourcing—a market which reaches an estimated tens of millions of people annually—and think not just about the technology that makes it possible, but also about the human workers and how they are impacted.

That author, Lilly Irani, a researcher and Ph.D. candidate in the Department of Informatics at the University of California Irvine, also developed Turkopticon, a software program released in February 2009 that was designed "as an ethically motivated response to crowdsourcing workers' invisibility," says Irani.

She recalls being troubled by what she perceived as the plight of workers being paid just a few dollars an hour to perform tasks on Amazon Mechanical Turk (MTurk), an online crowdsourcing marketplace launched by Amazon.com in November 2005 as a meeting place for "requesters" (employers) with large volumes of microtasks or HITs (Human Intelligence Tasks). Indeed, one worker complaint heard over and over again was that requestors are able to walk away with the work submitted without paying for it, because Amazon leaves payment completely up to the discretion of the employers, who can claim they are unhappy with the quality of the work.

"There is no process for the employer to justify its decisions to workers or to Amazon," she says.

PHOTOGRAPH FROM SHUTTERSTOCK.COM

And so Irani created Turkopticon, a software tool cheekily named after a prison surveillance design with a guard tower in which there may or may not be a guard. The *possibility* of surveillance induces prisoners to discipline themselves, says Irani.

"Our hope is that Turkopticon will not only hold employers accountable, but also induce better behavior."

Functionally, Turkopticon is a browser extension that, when workers search MTurk for HITs, "scrapes the requester ID that is within the HITs market page list and inserts any reviews that other workers have written about that particular requester. So that when a worker is deciding whether they want to take the assignment or not, they can also review a quick summary of what other workers have said. Our hope is that Turkopticon will not only hold employers accountable, but also induce better behavior," she explains.

Turkopticon reportedly gets 120,000 page views per month and has been installed almost 10,000 times. Yet, has it made a difference?

"I can't say that crowdsource wages have gone up," Irani says. "But I've heard requesters say at crowdsourcing conferences that it's become important for them to establish good reputations with workers. If they put out tasks that aren't clear or if they upset workers by not paying them, they sometimes need to create a new account and start from scratch. They find that treating workers badly can certainly raise the cost of doing business."

Luis von Ahn, a professor of computer science at Carnegie Mellon University and a crowdsourcing expert, says Irani is one of the few people trying to stop abuse on the part of the employers. Much more frequently, the abuse that takes place is on the part of the workers who try to game the system, hoping to get paid for little or no effort, he observes.

"Imagine a crowdsource task that pays people to look at images and tag each one with a description," he explains. "A worker can just, say, hit the 'F' key a few times and hope to get paid for that useless input."

Employers use various techniques to limit such shenanigans—either by refusing to pay for a response unless at least one other worker agrees with that response, or by testing each worker to make sure they are capable of doing the task.

Von Ahn estimates that, on Mechanical Turk, 10%–20% of the workers try to cheat in some form or another. Far fewer employers cheat, he maintains.

"We're talking about workers who may be in some other country, often India, who can easily remain anonymous, and so it's easy for them to cheat," von Ahn explains. "On the other hand, the employer is usually a university or a large company like Google and is much less prone to cheating because they have a lot more to lose."

Von Ahn does agree with Irani on one count: "She built Turkopticon to protect the little guy, and that's a good thing, I think," he says. "When the little guy, the worker, gets cheated out of the buck or two per hour that he makes—and that is what most of these crowdsource workers do make—that's a lot nastier than if the employer loses the money. Because a few bucks mean a lot more to the guy who lives in India than to the employer."

Anand Kulkarni agrees. He is the CEO of MobileWorks, a competitor to MTurk that bills itself as an online labor platform designed to put the workers' interests first.

"MobileWorks is predicated on the idea that if we pay workers more and let them work under their real names in a collaborative environment that is more similar to a real-world workplace, they will perform better and deliver better-quality results compared to anonymous online work systems that are balanced against the workers," he explains.

MobileWorks charges its small business and corporate clients that make use of the crowd directly for its services and for use of the crowd, says Kulkarni. "We shield them from the complexity of interacting with the crowd themselves and we charge for doing so."

He compares elements of what MobileWorks does to "a digital union of sorts." When a task is posted, his team ensures it carries a "meaningful price based on the prevailing wages in the location of the workers wanted." And they make sure the workers get paid if the work has been completed.

Workers who join the platform are trained, receive certifications for certain skills, and then MobileWorks vouches for them and helps them graduate into the broader economy where they can find jobs, either through MobileWorks or on other sites.

As in Turkopticon, MobileWorks workers are encouraged to provide feedback on requesters.

Kulkarni believes that, despite efforts to eliminate abuses on MTurk, they continue to exist. On the other hand, he says, more and more alternative crowdsourcing systems are making efforts to develop solutions to these problems.

"On MobileWorks, we explicitly prevent these kinds of abuses by employers," he says, "while others—like Elance.com, Freelancer.com, and oDesk.com—have built-in mechanisms to mediate between employers and employees."

ACM Member News

NANCY AMATO IS PASSIONATE ABOUT RESEARCH, MENTORING

Texas A&M University computer science and engineering professor Nancy Amato is passionate about two things: her research on motion planning, robotics, and computational biology, and her mentoring of a diverse group of students.

The Pacific Northwest native, who refers to herself as an "accidental computer scientist" because it took her so long to decide on a career course, received undergraduate degrees in Mathematical Sciences and Economics from Stanford University, and M.S. and Ph.D. degrees in computer science from the University of California at Berkeley and the University of Illinois at Urbana-Champaign.

She co-directs Texas A&M's Parasol Lab, is a deputy director of the Institute for Applied Math and Computational Science, an associate director of the Center for Large-Scale Scientific Simulations, chair of the university-level Alliance for Bioinformatics, Computational Biology, and Systems Biology, and is past chair of the Council of Principle Investigators (2009–2010).

Striving for practical application of her research, Amato works on motion-planning problems in abstract settings and applies them to real-world situations like robotic surgeries and architectural applications. She currently is using motion-planning algorithms to devise better methods of evacuation planning. "Our goal is to modify architectural designs to enable first responders to speed emergency evacuations," Amato explains.

Amato serves as co-director for the Distributed Research Experience for Undergraduates mentoring program, a joint program of the Computer Research Association's Committee on the Status of Women in Computing Research and the Coalition to Diversify Computing. In 2012, Amato accepted Texas high school senior Kensen Shi into the program. Shi, working with Amato's graduate students, wrote a paper on a more-efficient motion-planning algorithm; he subsequently entered the 2012 Siemens Competition in Math, Science and Technology and won first prize, which includes a $100,000 scholarship. "Kensen is brilliant and I felt very proud to have mentored him; in a few years I'll be saying, 'I knew him when.'"

—Laura DiDio

"Suddenly, when you need a whole bunch of people to do something useful, platforms like ours let you organize them very well."

Still, not everyone in the crowdsourcing space believes abuses are a major concern.

At CrowdFlower, founder and CEO Lukas Biewald observes that crowdsourcing gives people the opportunity to choose those tasks they want to perform when they want to perform them, and earn money or coupons or in-game currency for completing those tasks.

"That seems like a good deal to me," he says. "We have over four million people working on CrowdFlower syndicated tasks, and I log into our partner sites all the time to make sure that it's a good experience for everyone. There are bound to be some complaints and some issues, but we work as hard as we can to resolve the issues as quickly as possible."

CrowdFlower's business model is different from that of MobileWorks; instead of posting jobs on its own site, it uses an API to syndicate tasks from customers and then farm them out to thousands of partner sites, one of them being MTurk.

Like Turkopticon and MobileWorks, CrowdFlower uses technology—but mainly to guarantee the quality of the work for the employer.

Biewald explains that having a background in AI enabled him write the statistical algorithms that predict the likelihood that a task will be done correctly—and how many checks and redundancies need to be built into the tasks to produce optimal results.

"We have the work history of the people who sign on," he says, "so many of our decisions are based on their historical performance. For instance, if we think someone is going to be 90% accurate and the customer requires 99% accuracy, we know we need to have a second person check the work."

Also, he says, there are other indicators—if a person works too quickly or has never done a similar task before, the software sends up a flag.

Biewald believes few workers are dissatisfied, and the focus of those exploring crowdsourcing should be on the good it does, not just for businesses but also for the general population.

"I'm sure you can find someone, somewhere, who didn't have a good experience," he says, "but if you interview workers—and I do that all the time—they are mostly really happy. I mean, the system wouldn't work if people didn't want to work for us."

He relates one task that began shortly after the 2010 earthquake in Haiti. The U.S. State Department had set up a number to which Haitians could send text messages to report emergencies. As it turned out, the messages were in Creole and, to make matters worse, were written in text message slang that prevented Google from translating them. Because the State Department did not have enough bilingual people to read through the huge volume of messages it was receiving, it contracted with CrowdFlower to seek out crowdworkers who spoke Creole and could translate—and they did so on a voluntary basis.

"Suddenly, when you need a whole bunch of people to do something useful," says Biewald, "platforms like ours let you organize them very well."

So the debate continues—between those who believe the crowdsourcing technology is enabling employers to take advantage of workers...and those, like Biewald, who see crowdsourcing opening up new opportunities for employers and workers alike.

Indeed, says Biewald, the future of crowdsourcing is generally trending away from very simple tasks and toward higher-level specialized tasks.

Carnegie Mellon's von Ahn agrees, citing the Haitian scenario as just one example of what crowdsourcing can do, beyond having people tag photos and check websites for duplicate images. Another example, he says, is how crowdsourcing was used to translate tweets during the Arab Spring.

"Much of the discussion in the crowdsourcing space is by computer professionals and employers who are trying to make the work more efficient by making the workers more efficient and by making the tasks more exact," he says. "Perhaps there should be more focus on that, and less on how to get around the minimum wage laws." C

Further Reading

von Ahn, L.,
"Human Computation," (video), August 22, 2012, http://www.youtube.com/watch?v=tx082gDwGcM

Cushing, E.,
"Dawn Of The Digital Sweatshop," August 1, 2012, "East Bay Express," http://www.eastbayexpress.com/oakland/dawn-of-the-digital-sweatshop/Content?oid=3301022

Kittur, A., Nickerson, J., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., and Horton, J.,
"The Future of Crowd Work," published February 2013 at the 16th ACM Conference on Computer Supported Cooperative Work, http://hci.stanford.edu/publications/2013/CrowdWork/futureofcrowdwork-cscw2013.pdf

Neumann, E.,
"Tech Company CrowdFlower Denies Labor Violation," December 4, 2012, "MissionLocal," http://missionlocal.org/2012/12/tech-company-crowdflower-denies-underpaying-workers/

Ipeirotis, P.,
"Mechanical Turk: Now with 40.92% spam," December 16, 2010, http://www.behind-the-enemy-lines.com/2010/12/mechanical-turk-now-with-4092-spam.html

Irani, L., and Silverman, M.,
"Turkopticon: Interrupting Worker Invisibility in Amazon Mechanical Turk," April 2013, ACM SIGCHI Conference on Human Factors in Computing Systems, http://www.ics.uci.edu/~lirani/Irani-Silberman-Turkopticon-camready.pdf

Van Pelt, C., and Sorkin, A.,
"Designing a Scalable Crowdsourcing Platform," 2012 ACM SIGMOD International Conference on Management of Data, http://dl.acm.org/citation.cfm?id=2213951

Biewald, L.,
"Massive Multiplayer Human Computation for Fun, Money, and Survival," ICWE 2011, http://link.springer.com/chapter/10.1007%2F978-3-642-27997-3_18?LI=true#

Hester, V., Shaw, A,. and Biewald, L.,
"Scalable crisis relief: Crowdsourced SMS translation and categorization with Mission 4636," ACM Symposium on Computing for Development 2010, http://dl.acm.org/citation.cfm?id=1926199

Paul Hyman is a science and technology writer based in Great Neck, NY.

DOI:10.1145/2492007.2492016

Richard Heeks and Andrew Robinson

Emerging Markets

Ultra-Low-Cost Computing and Developing Countries

Raspberry Pi and its potential in the "global South."

ALONGSIDE THE STEADY progress of Moore's Law, we sometimes see step function changes in price-performance ratios of IT. One such change has produced ultra-low-cost computing (ULCC) over the past several years. In essence, this wraps computing peripherals around a cellphone hardware core; producing a computer for just a few tens of dollars. ULCC enables computing to reach applications and users that other computers cannot, and it is spinning out in multiple directions. In this column, we will focus on one ULCC development—Raspberry Pi—and one particular direction: use in developing countries.

Raspberry Pi was created to address the decline in numbers and skill levels of school leavers applying to study computer science. Eben Upton, founder of the non-profit Raspberry Pi Foundation, was an admissions tutor for Cambridge University. He felt university applicants lacked the experience of experimenting and programming that was common in the 1980s and 1990s.

In those earlier decades, the flashing cursor interface made users creative from the start and provided a direct opportunity for programming. But steady commoditization had led IT to become closed and locked down. For modern schoolkids, hardware meant sealed units of brushed aluminum and glossy plastic; software meant apps that were to be used but not understood.

Pi was therefore a "back to the future" idea; trying to open up IT in all possible ways, and to get young people tinkering with all aspects of the technology. To do that, it needed two things: a very low price (targeting $35 for the complete Model B and $25 for the cut-down Model A) and an open design.

In hardware terms, low cost was possible thanks to advances in integration that have effectively shrunk all the components of a desktop computer into a single silicon chip. The Raspberry Pi is based around a 700MHz ARM11 system on chip (SOC) with a powerful graphics co-processor. Typically this sort of processor was used in cellphones five years ago. Stacked on top of the processor is 512MB RAM on Model B and 256MB on Model A.

The SOC drives a HDMI output allowing high-definition display on the latest screens, and composite video out as used for monitors since the 1980s. The Model B has two USB sockets and an Ethernet connection, requiring less than 5W of power, with Model A (having only one USB socket) requiring 2W. Power is provided via a micro USB connector, allowing cellphone chargers or any 5V supply (including batteries, solar panels, or other renewable sources) to be used.

Apart from the graphics processor, which is propriety to Broadcom, the Raspberry Pi is completely open source, which helps to keep costs down. From the circuit schematics to the applications and the operating system, anyone can examine and contribute online. The Foundation provides a version of Debian Linux that presents users with a basic text login rather than

Students using Raspberry Pi-based computers in a village school in Cameroon.

PHOTOGRAPH BY GEERT MAERTENS

a slick GUI by default, with the entire operating system and user files stored on a swappable SD card.

Users can "see into" the underlying mechanics of the technology—quite literally since the basic Pi comes as just a case-less board. Users must build their own computer by connecting monitor, keyboard, mouse, and other peripherals to the motherboard; and load their own operating system and applications if they want to word process, web browse, and so forth. Open design and an open innovation approach have encouraged a whole set of further developments, such as a hardware interface (PiFace) that can connect the computer to a variety of sensors and peripherals.

But can Pi break out of its computer science teaching and design project ghetto to address the needs of developing countries?

ULCC's promise is that it will match the explosion in digital communications with an explosion in data processing capacity—bringing the "I" of information and communication technologies to the mass of the world's population just as cellphones are bringing the "C."

As we noted in a recent blog,[1] three application opportunities come to mind:

▸ **Micro-enterprise and household computing**: providing access to standard computing applications for the individual enterprise and household. Add a mobile Internet connection, and we might finally move beyond the "telecenter" model of community computing, to something much more integral to the lives of those on lowest incomes.

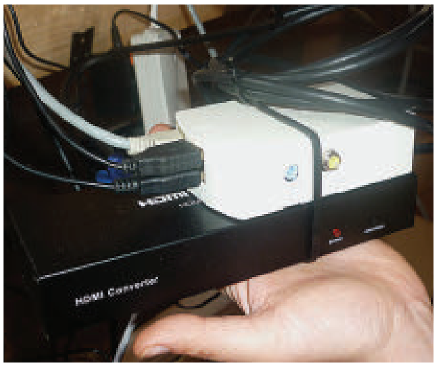
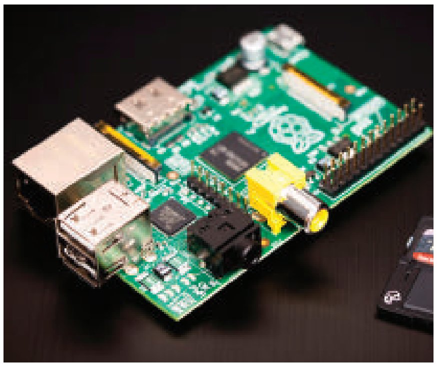
▸ **Technical education**: as noted earlier, the prime motivation behind Pi was to reignite interest in computing as a subject among schoolchildren. There is a great thirst for IT education in schools, colleges, and universities in developing countries but budgetary constraints are a major barrier. Pi might help to overcome those but also allow students to open the box and play about much more, learning how IT works. A few projects have begun, such as the recent equipping of a village school in Cameroon.[3]

The spread of computing applications for the poor has been limited by design-reality gaps.

▸ **Data collection and automation applications**: there's a trickle of new electronic applications for development—smart motor controllers that save power and extend motor life, low-cost health monitors, water quality and climate change measurement devices, field-based agricultural sensors. Raspberry Pi might turn that trickle into a steady stream.

At root, though, the transformational promise of ULCC may not be about a specific application, but about a new approach to innovation and design. The spread of computing applications for the poor has been limited by design-reality gaps. Applications are developed by those external to poor communities, with designs that mismatch local realities, and thus often fail.

If ULCCs can become widespread, they can enable a new computing design paradigm: grassroots innovation in which local users are also designers; creating designs that match local needs, resources, and context. ULCC will also allow a new model of collaborative IT innovation: working alongside base-of-the-pyramid users. Large firms, university departments, and social enterprises could now afford rapid, mass prototyping—trying out and iterating quickly through many different models until they find one that works.

Raspberry Pi with peripherals attached (left) and the Raspberry Pi Model B (right).

But will this development opportunity be realized? Recent history is littered with the skeletons of low-cost computing-for-development failures such as the People's PC and the Simputer.[a] Question marks continue to be raised over the One-Laptop-Per-Child initiative.[4] So will ULCC and Raspberry Pi be any different?

Drawing on recent work analyzing what makes IT innovations scale in low-income markets,[2] we can identify three domains that need to function effectively:

Supply factors. Per-computer costs in the Cameroon school were above $300, nearly three-quarters of which was the cost of monitors and monitor HDMI/VGA convertors. Sourcing or development of low-cost monitors will be a necessity if ULCC is to be the basis for PCs as opposed to the controllers/sensors of the third opportunity—data collection and automatic applications—described earlier.

Production and local distribution and marketing will also need to be addressed; something on which the Raspberry Pi Foundation is playing catch-up. It was formed as a spare-time, pro bono initiative that expected to ship 10,000 units in total, and which has struggled to meet demand in the global North that is already around the one-million mark; leave alone thoughts about addressing the global South. Perhaps the best hope is that it might, as the One-Laptop-Per-Child initiative did, induce copycat private manufacturers to follow suit.

Demand factors. If, ironically, cost is not yet the unique selling point for ULCC, then what is? Likely, it is "tinker-ability" but that will limit demand to college and university sites in the developing world. Households, communities, even most schools do not want to tinker; they want something that will deliver useful applications. For them Pi in its current form may well be underdesigned and underbundled: their demand is for something more like a current smartphone or netbook. Ultra-low-cost computing will only find a major market if it can deliver those types of devices at significantly lower cost.

The alternative is, once again, the data collection and automation applications opportunity: the "gizmo route" that would find a killer application for the developing world in an electronic device built around ULCC; a device that would meet an important need of low-income communities. If that does emerge, it will come most likely from the grassroots or collaborative innovation approaches described.

Contextual factors. Even if these supply and demand factors can be met, ULCC in developing countries faces other challenges. Poor access to electricity probably is not one. This remains a major computing-for-development stumbling block: it tripped up efforts to use biometrics in the 2013 Kenyan elections. But ULCC devices' very low power consumption means they can be used much more easily in off-grid environments.

The limited skill infrastructure is much more problematic. Underbundling demands greater skills at the point of implementation, and ongoing literacy and IT skill deficits remain significant generic barriers to computing. Yet again, packaging ultra-low-cost computing into realtively simple electronic devices may be the answer.

a Acknowledgment to John L. King for this and a number of other points in this column.

Conclusion

In sum, ultra-low-cost computing offers a good amount of promise for development but that is true of most IT innovations. The core issue is what lies between that promise and widespread use. At the moment, the challenges to scaling ULCC for people at the base-of-the-pyramid look daunting. Raspberry Pi, at least, may remain a tertiary education niche product with a very few secondary education implementations.

One cannot help remembering, though, that mobile telephony looked very much like this at the end of the 1990s. Firms saw no demand among low-income users and had no plans to develop that market. Only after a few innovators stepped in—including nonprofits and international donors—was there a demonstration effect that induced larger players to enter.

We will wait and see if there is any such ULCC demonstration effect, either for computers or for electronic devices. At present, this is a blank canvas waiting to be painted, held back by our need to reconceptualize; to rethink what is possible and what is feasible and what is desirable in a world of very cheap computing power.

In part the next chapter will depend on many of those within the ACM community: Can we reconceptualize and develop new ideas, initiatives, and partnerships that will fulfill ULCC's socioeconomic development potential? Over to you... C

References

1. Heeks, R. Raspberry Pi: A paradigm shift for ICT4D? *ICTs for Development.* (Oct. 29, 2012); http://ict4dblog.wordpress.com/2012/10/29/raspberry-pi-a-paradigm-shift-for-ict4d/
2. Heeks, R. Why M-Pesa outperforms other developing country mobile money schemes. *ICTs for Development* (Nov. 24, 2012); http://ict4dblog.wordpress.com/2012/11/24/why-m-pesa-outperforms-other-developing-country-mobile-money-schemes/
3. Maertens, G. Bringing computing to rural Cameroon. *Raspberry Pi* (Apr. 4, 2013); http://www.raspberrypi.org/archives/3634
4. Ozler, B. One Laptop Per Child is not improving reading or math. *Development Impact* (June 14, 2012); http://blogs.worldbank.org/impactevaluations/one-laptop-per-child-is-not-improving-reading-or-math-but-are-we-learning-enough-from-these-evaluati

Richard Heeks (richard.heeks@manchester.ac.uk) is Director of the Centre for Development Informatics at the University of Manchester, U.K.; http://www.cdi.manchester.ac.uk/.

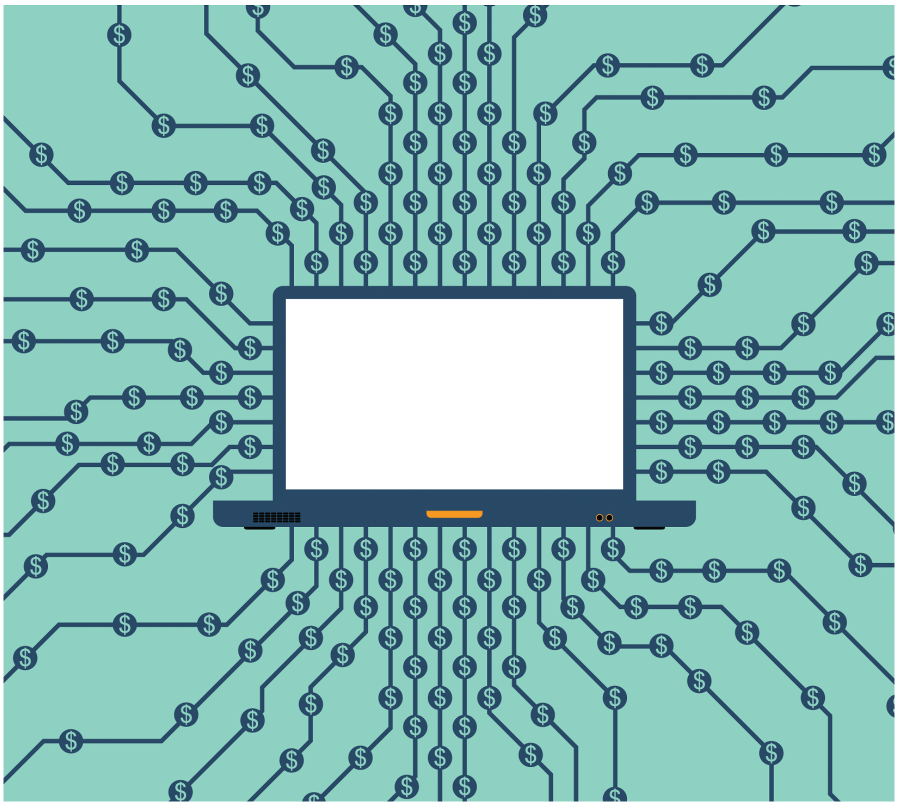
Andrew Robinson (andrew.robinson@cs.man.ac.uk) has worked in education and public outreach at the School of Computer Science at the University of Manchester, U.K.

DOI:10.1145/2492007.2492017

Chrysanthos Dellarocas and Marshall Van Alstyne

Economic and Business Dimensions
Money Models for MOOCs

Considering new business models for massive open online courses.



DESPITE THE MASSIVE media ink spilled over massive open online courses, the ink spilled by MOOCs themselves remains red. MOOCs lose money. Most are free.[3] Universities and venture capitalists subsidize them while searching for the class of the future. This cannot continue but their future, we believe, is bright.

Education is only the latest industry to face digital disruption. Music, movies, news, travel and real estate already traveled this path. Conventional business models—charging customers directly for products and services—are often ineffective online. Media companies painfully discovered that free alternatives such as YouTube videos, news blogs, independent fiction, Wikipedia pages, and the ease of piracy place limits on charging for content. Travel agencies discovered, equally painfully, that free alternatives and consumer ratings place limits on charging for bookings and advice.

After years of trying to replicate old business models online, companies, or their competitors, built platforms that offer free service and information as bait to attract users and their activity. These platforms monetize eyeballs, comments, referrals, and relationships based on two key ideas:

▸ *Charge for complements*, including analytics and value-adding activities performed by users.[2] Red Hat Linux offers Linux software for free and charges for consulting and technical support. Tumblr offers blogging and social networking for free and charges for analytics. From a MOOC's perspective, teaching a man to fish allows us to sell him a boat. We can also sell the fish he caught while learning.

▸ *Charge a different group with interdependent demand.*[6,7] TripAdvisor offers free advice to travelers and charges airlines and hotels. LinkedIn offers many free services to job seekers and charges recruiters. Teaching a man to fish, we can charge fleet captains who hire him.

The first idea defines what one pays for, which can be content or complement; the second idea specifies who pays. We used these ideas to create a matrix of possible business models, shown in the accompanying table, and identified a number of plausible money models for MOOCs.[a] We organize our discussion by who pays.

a We certainly do not claim our current list is complete and we invite readers to populate cells we left blank with interesting new ideas.

ILLUSTRATION BY HERMIN UTOMO

States

State Subsidies. Most countries subsidize education as a key state function. The case for subsidizing education based on socially interdependent demand could hardly be stronger. People who complete high school pay more taxes, vote more often, volunteer more often, have higher savings rates that stimulate more investment; commit fewer violent crimes, live longer, use less health care, and consume fewer welfare services.[1] MOOCs could be as defensible as Pell grants. This is already happening. Based on San Jose State University's experiments with edX and Udacity, the state of California plans to expand use of MOOCs to other state campuses.[4]

A framework for organizing MOOC business models.

	What are they paying for?			
Who pays?	*Course Content*	*Data and Analytics*	*Platform Activity (Student labor)*	*Complementary Services*
States	State subsidies			
Students	Tuition	Diagnostics	Peer Assistance	Certification Tutoring Collaborative group learning
Employers	Custom courses Continuing education	Recruiting, Analytics		Certification
Sponsors	Sponsored courses		Problem-sponsored learning	Access to experts
Other Platforms	Syndicated courses	Student recruiting services		

Students

Tuition. We believe education is better positioned than media to generate direct revenue from content. In contrast to music, film, and books, piracy is less likely to limit MOOC charging ability. Digital courses are interactive services whose completion often requires live coordination with other students, graders, and server-based staff. One could watch pirated lectures and work through pirated problems, but it takes the participation of others to answer questions, grade assignments, and ultimately verify completion. Each of these represents a control point ensuring the platform receives payment.[b] Given pricing power, charges can vary with business and pleasure: degree classes cost more, avocational classes cost less.

In addition to tuition models, MOOCs can also charge based on a variety of *freemium* models, meaning a business logic where basic course content is free and students pay for complements as with the following examples:

Certification signals to others that a student has mastered course content. Coursera is already experimenting with the idea of making MOOCs available for free but charging for credentialing. One level could certify completion and another could certify skill. The University of Washington, a Coursera partner, is testing a hybrid model of a free MOOC offered simultaneously with more academically rigorous credit-bearing version that includes a fee. Udacity is partnering with Pearson's extensive network of testing centers to offer similar, fee-based, certification services.

Diagnostics. MOOC platforms can use rich data generated from online interaction to offer personalized diagnostic analytics that identify student strengths and weaknesses and adjust pace of delivery to match. These capabilities can be offered as value-adding fee-based services or extra-credit practice.

Tutoring and Peer Assistance. When students flounder, the MOOC can offer online "genius bars" staffed by faculty experts and accredited students from previous classes, for a fee.[c] Tutors and peers can answer questions while helping push students to higher achievement.

Collaborative Group Learning. MOOCs have atrocious attrition rates. One solution is to charge students for recommended study groups to help find compatible peers. Another option borrows from "Grameen Banking" where people apply for loans as a team (see http://en.wikipedia.org/wiki/Grameen_Bank). MOOC groups could self-organize and commit to learning like the teams that take out loans together. Study groups solve both the adverse selection problem (bad risk loans/uncommitted students) and the moral hazard problem (ex post monitoring/letting your peers down). Pricing in such schemes can be a simple up-front payment with a rebate for completing with one's peers.

Employers

Recruiting services. To compete on talent, companies can either hire more talented workers or improve the talent they have. MOOC funding models thus have opportunities both pre- and post-employment. To help employers tap into new talent, MOOC *analytics* can provide information on student qualifications and improve the hiring process. Digital learning platforms generate substantially more detailed insights into prospective employee behaviors than transcripts. Firms can also recruit the best students before they enter the job market.

Udacity has been running a recruiters program using its database of students to identify candidates who would be good matches for openings at partner companies like Google, Amazon, and Facebook among others. The MOOC platform, matching inter-

MOOCs could be as defensible as Pell grants.

b Of course, students might acquire skills using pirated materials then obtain credentials from third-party agencies. Pricing decisions must reflect competition in the same manner that iTunes pricing has curbed music piracy. Reputation effects can also make such students unpopular with employers.

c We thank Jeff Jarvis for this idea.

dependent student and employer profiles, can mediate access for a fee.

Custom courses. Despite high youth unemployment, many businesses complain of a skills shortage in key areas and of a mismatch between what companies need and what academic institutions produce.[5] Businesses can commission MOOCs or specific courses tailored to their requirements. These subsidies can serve both marketing and recruiting purposes. For example, Google could commission a "branded" MOOC on software engineering, with very challenging assignments and exams, carrying a promise that top performers automatically receive internships.

Continuing education. Giving existing workers new skills can be cheaper than hiring new workers. Many Fortune 500 companies support workers' interests in pursuing additional training via matching funds, time off, and tuition rebates. The low cost and scheduling flexibility of MOOCs makes them very strong candidates for support.

Sponsors

Sponsored courses. Having attracted vast numbers of "edsumers,"[d] advertising is an obvious option. Google, Amazon, LinkedIn and professional societies are well positioned to offer or cross-subsidize education. Traditional advertising, however, can distract from learning and classier mechanisms can succeed. Georgia Institute of Technology and AT&T have launched the first online, university certified, degree in computer science. Not only does AT&T gain access to well-trained programmers but online education also increases demand for *complementary* telecommunications infrastructure. Broadband replaces traditional capital stock.

Access to experts. Inspired by record labels, another *complements* business model can pursue the lucrative sales associated with star professors. Under "360" or full rights contracts, the big recording labels invest in artist promotion and up-front marketing in exchange for a percentage of artists' concert and merchandizing revenues. Record labels have expanded their view of assets from the content they produce to the star performers they access. Although lecturers have not traditionally behaved like performers, MOOCs have the potential to create global superstars who generate revenue through speaking engagements, expert witness testimony, and consulting. MOOC platforms could thus double as speakers' bureaus or expert agencies.

Digital learning platforms generate substantially more detailed insights into prospective employee behaviors than transcripts.

Problem-Sponsored Learning. One interesting new idea blends user-generated content with experiential learning, a business model we call *Problem-Sponsored Learning*. Organizations facing important challenges can sponsor student projects. As with commissioned works in Renaissance art studios, apprentices can solve problems under the guidance of expert mentors. In the context of MOOCs, tutoring can be partially automated and students can learn from each other. At the scale of thousands, the best solutions should be excellent while the inferior solutions are no longer waste but opportunities for learning. One can imagine even non-profit organizations being able to afford code, graphic design, tax preparation, advertising copy, or data analytics at affordable prices. Patronage has historically sponsored art, music, chemistry, poetry, philosophy, and more.

A modern day stepping-stone to this idea is Innocentive, which operates a platform that matches companies who have problems to people who have solutions. Innocentive organizes online contests where the spon-

d Compliments to Michael Schrage.

Calendar of Events

August 12–16
ACM SIGCOMM 2013 Conference, Hong Kong, China, Sponsored: SIGCOMM, Contact: Jia Wang, Email: jiawang@research.att.com

August 18–26
Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Saint Petersburg, Russia, Sponsored: SIGSOFT, Contact: Bertrand Meyer, Email: bertrand.meyer@inf.ethz.ch

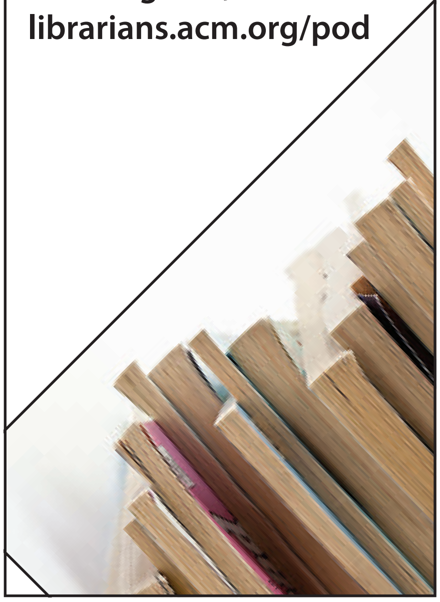
August 22–23
ACM Symposium on Applied Perception 2013, Dublin, Ireland, Sponsored: SIGGRAPH, Contact: Ludovic Hoyet, Email: lhoyet@gmail.com

August 25–29
Advances in Social Networks Analysis and Mining 2013, Niagara, Canada, Sponsored: SIGKDD, Contact: Jon Rokne, Email: rokne@ucalgary.ca

September 9–12
The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland, Sponsored: SIGCHI, SIGMOBILE, Contact: Silvia Santini, Email: santinis@wsn.tu-darmstadt.de

September 15–18
20th European MPI Users' Group Meeting, Madrid, Spain, Contact: Jack Dongarra, Email: dongarra@cs.utk.edu

October 6–9
The ACM International Conference on Interactive Tabletops and Surfaces, St. Andrews, UK, Sponsored: SIGCHI, Contact: Ashley Cozzi, Email: acozzi@acm.org



We foresee digital course syndication emerging in higher education.

soring company details their problem and offers a reward. The sponsor selects the preferred solutions, the winners receive rewards, and Innocentive receives a commission. Blending the Innocentive model with experiential learning gives students the opportunity to solve concrete problems and learn new and diverse concepts, as they progress toward mastering new skills. Their final output is also socially useful.

Other Platforms

Syndication. By 1846, the newspaper industry had discovered it makes no sense for every newspaper to write original articles for every newsworthy story. No single newspaper has the resources nor can any newspaper claim to be most qualified on all topics. The Associated Press introduced the practice of syndication as an obvious solution: member newspapers make their original content available to other members for a fee. In such associations, usually the best-qualified member produces content on a topic and everyone else licenses it, saving production costs and resulting in higher quality.

Today's traditional universities operate like newspapers that insist on producing original articles of variable quality for all stories. In a world of rising costs and global information transparency this is unsustainable. From an economic standpoint, universities will find it too costly to hire professors for every subject students request. From a quality perspective, students will steer clear of substandard local teaching when they know a blockbuster digital course is available.

We foresee digital course syndication emerging in higher education: Universities will form consortia (platforms) where they will each contribute their best digital courses, making them available to other members for a fee. Such courses will not necessarily be MOOCs, but, more likely, blended format courses involving local study groups at each participating university.

A startup company called 2U is already experimenting with this business model. 2U is forming a consortium of universities that are, each contributing blended online courses, to be potentially licensed and used by other consortium members. Current 2U members include Boston College, Emory, Notre Dame, Northwestern, and Washington University in St. Louis.

Student recruiting analytics. Traditional universities can become clients of MOOC analytics. As high school students vie for admission to colleges and undergraduates vie for admission to graduate programs, MOOCs can offer valuable data that identify and assess outstanding prospects. In California, for example, the college system often feeds its best students into the university system.

Academia must learn from what writer Jeff Jarvis calls "shovelware," old products foist on consumers in new media. We are very optimistic. Digital courseware will produce new business models and enormous social value in our increasingly connected world. C

References

1. Alliance for Excellent Education. The high cost of high school dropouts—What the nation pays for inadequate high schools. Issue Brief, Washington, D.C., 2007.
2. Cameron, L. and Bazelon, C. The impact of digitization on business models in copyright-driven industries: A review of the economic issues. The Brattle Group, Inc., 2013; http://brattle.com/_documents/UploadLibrary/Upload951.pdf.
3. Cusumano, M. Are the costs of 'free' too high in online education? *Commun. ACM 56*, 4 (Apr. 2012).
4. Lewin, T. and Markov, J. California to give Web courses a big trial. *The New York Times* (Jan. 15, 2013).
5. Mourshed, M., Farrell, D., and Barton, D. *Education to Employment: Designing a System that Works.* McKinsey Center for Government, 2013.
6. Parker, G., and Van Alstyne, M. Internetwork externalities and free information goods. In *Proceedings of the 2nd ACM Conference on Electronic Commerce.* ACM, 2000.
7. Parker, G., and M. Van Alstyne. Two-sided network effects: A theory of information product design. *Management Science 51*, 10 (Oct. 2005), 1494–1504.

Chrysanthos Dellarocas (dell@bu.edu) is a professor and the chair of the department of management information systems at Boston University.

Marshall Van Alstyne (mva@bu.edu) is an associate professor in the department of management information systems at Boston University and a research scientist at the MIT Center for Digital Business.

DOI:10.1145/2492007.2492018

Eric Byres

Privacy and Security

The Air Gap: SCADA's Enduring Security Myth

Attempting to use isolation as a security strategy for critical systems is unrealistic in an increasingly connected world.

As a security practitioner and a controls engineer, I am often asked my views on air gaps as a security strategy for supervisory control and data acquisition (SCADA) and industrial control systems (ICS). Air gaps have long been a focus of discussion in industry, and they still continue to generate a lot of interest in the media. In theory, the air gap strategy certainly sounds great. By creating a physical gap between the control network and the business network, the bad guys—criminals, hackers, and worms—are kept out of critical systems.

Before I go any further, I must clarify what I mean when I use the term "air gap": What I am referring to in this column is the philosophy that says we can truly isolate our critical systems from the outside world. And this is where the myth—and the danger—lies. To begin, I do not believe true air gaps actually exist in the ICS and SCADA world. Moreover, many SCADA security experts have even stronger opinions than me on the subject—for example, see Craig Wright's blog.[a] However, I do acknowledge (albeit reluctantly) that not everyone agrees with me on this.

In 2011, for example, we saw a deluge of SCADA and ICS vulnerability notices with advice on addressing the issue by using an air gap. One example I have referred to in the past comes from the original Siemens Security Advisory addressing the vulnerabilities in Siemens' SIMATIC S7-1200 PLC line: "In addition, it is important to ensure your automation network is protected from unauthorized access using the strategies suggested in this document or isolate the automation network from all other networks using an air gap."

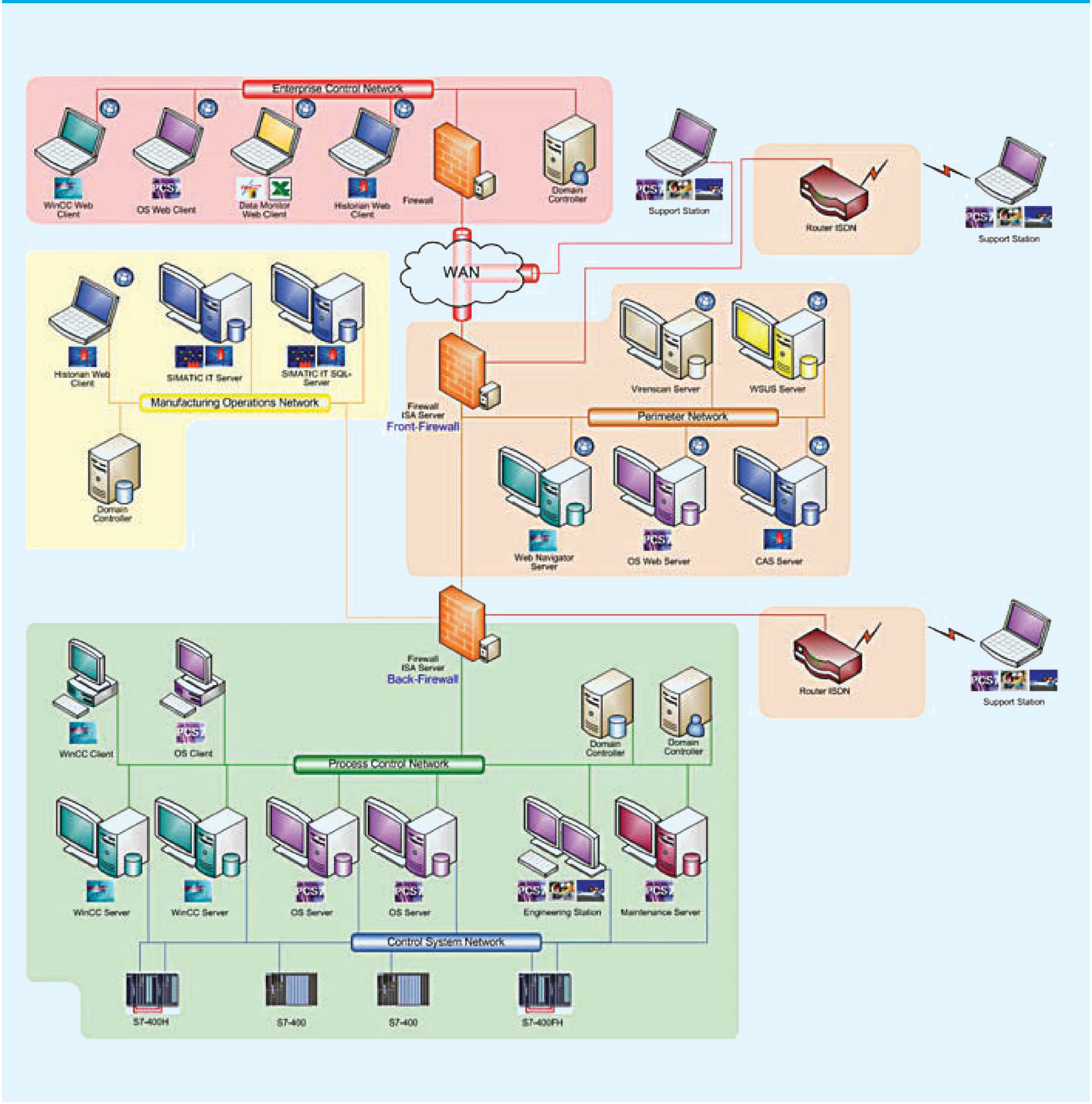
A control system protected by a real air gap: IBM 360/195 playing chess, November 1974.

The 'Air Gap Principle' Is History

To give credit where credit is due: Siemens removed this recommendation from the advisory (and all

a http://infosecisland.com/blogview/16770-SCADA-Air-Gaps-Do-Not-Exist.html

A high-security architecture.



other advisories) a few months later. I strongly suspect that Stefan Woronka, Siemens' director of Industrial Security Services, had something to do with this when he publicly stated: "Forget the myth of the air gap—the control system that is completely isolated is history."

Similarly, all the security advisories from two other leading vendors (Schneider Electric and Rockwell) make no mention of air gaps. Rockwell's mitigation guidance is very clear: "Block all traffic to the EtherNet/IP or other CIP protocol-based devices from outside the Manufacturing Zone by restricting or blocking access to TCP and UDP Port#2222 and Port#44818 using appropriate security technology (for example, a firewall, UTM devices, or other security appliance)."[b]

Could this be an indication that control system vendors are beginning to realize air gaps conflict with their architectures? For example, consider the accompanying figure diagramming a high-security architecture derived from the Siemens' Security Concept manual.[c] Can you spot the air gap in the figure? I can't!

Are you ready for another challenge? Try this exercise:

b Source: KB Article 470154-EtherNet/IP™ Product Vulnerabilities.

c See http://cache.automation.siemens.com/dnl/jE/jE2MjIwNQAA_26462131_HB/wp_sec_b.pdf.

▸ Download the security manual from Rockwell.[d]

▸ Search for the term "Air Gap." You won't find it.

▸ Search the diagrams for an air gap. You won't find it.

▸ In fact, while you are at it, why not check out all the major SCADA vendors' engineering guides. You won't find the air gap mentioned anywhere (if you do find an example of an industrial vendor recommending air gaps, please send it to me).

Air Gaps Do Not Work in the Real World

There is a good reason why you will not find the air gap mentioned in vendor engineering manuals and why it is disappearing from security advisories. As a theory, the air gap is wonderful. In real life, it just does not work.

Sure, you can simply unplug the connection between the control system and the business network and presto, you have an "air gap." Excellent! Job done!

Then one day the bubble bursts. Your control system team gets new logic from the engineering consultant—perhaps it addresses a design flaw that has been causing your company considerable downtime… A little while later Adobe sends your team a software update—perhaps it is for a critical vulnerability in the PDF reader the staff uses to view operational manuals…Next the lab group sends a process recipe that will improve product quality. Are you starting to get the picture?

The list just keeps growing and growing—patches for critical computer operating systems, anti-virus signatures, remote support from vendors—no company can ignore them all.

So what does the average controls engineer do? Just load some files onto a USB flash drive and carry that onto the plant floor. But wait a minute—isn't that how Stuxnet spread?

Hmmm, let's see…maybe putting everything onto a laptop is the solution? Yes, that's the ticket! Oh, but what if the laptop is infected?

Eureka! A serial line and a modem! But wait a minute—the Slammer worm got into a number of control systems that way. Yes, even the trusty old CD can be turned into the carrier of evil bits.

Clearly, it is time for the media, consultants, and end users to give up on the air gap myth.

As much as we want to pretend otherwise, modern industrial control systems need a steady diet of electronic information from the outside world. Severing the network connection with an air gap simply spawns new pathways like the mobile laptop and the USB flash drive, which are more difficult to manage and just as easy to infect.

Air Gaps Do Exist In Trivial and Very High Risk Control Systems

So are there air gaps in any control systems? Sure: one example appears in the photograph on the first page of this column. For another, more real-world, example: the digital thermostat controlling the heat pump in my home probably has a true air gap. And maybe in extremely high-risk systems—I am led to believe reactor control systems in nuclear plants are truly air gapped.

But do air gaps exist for all the control systems that manage our power grid, our transportation systems, our water systems, and our factories? Consider how Sean McGurk, the former director of National Cybersecurity and Communications Integration Center (NCCIC) at the U.S. Department of Homeland Security answered that question: "In our experience in conducting hundreds of vulnerability assessments in the private sector, in no case have we ever found the operations network, the SCADA system, or energy management system separated from the enterprise network. On average, we see 11 direct connections between those networks. In some extreme cases, we have identified up to 250 connections between the actual producing network and the enterprise network."[e]

The End of the Fairy Tale—Time for Industry to Grow Up

For many years, control system vendors have believed (or wanted to believe) in the fairy tale of the air gap. Now they have grown up and have come to realize this security strategy is finished. The government agencies like ICS-CERT have also accepted that a true air gap is impossible.

All control systems are connected to the outside world in some fashion. It might be a network connection, a serial line, or USB flash drive "sneakernet," but it is a pathway that can be exploited by modern malware like Stuxnet and Flame. Cyber security countermeasures must face up to this fact.

Clearly, it is time for the media, consultants, and end users to give up on the air gap myth. Believing a critical SCADA system's security is under control because it is "isolated" is just a dangerous illusion. As stated by Chris Blask, CEO of ICS Cybersecurity, Inc.: "None of the vulnerabilities [uncovered at the NESCOR summit] pose as great a risk as the belief that your system is isolated."

Any company defending its critical SCADA systems with an air gap is making a serious mistake. Any security consultant recommending air gaps as a strategy is doing their client a serious disservice. And any vendor suggesting air gaps as a solution to their product vulnerabilities is being irresponsible. It is time we put the air gap on the shelf with other fairy tales and started designing real-world solutions to protect the critical SCADA systems running our world. C

d See http://literature.rockwellautomation.com/idc/groups/literature/documents/wp/enet-wp005_-en-e.pdf.

e Source: The Subcommittee on National Security, Homeland Defense, and Foreign Operations May 25, 2011 hearing.

Eric Byres (eric.byres@belden.com) is the chief technology officer at Tofino Security in British Columbia, Canada, and a member of the ISA and IEC committees for control system security.

DOI:10.1145/2492007.2492019

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
Cherry-Picking and the Scientific Method

Software is supposed be a part of computer science, and science demands proof.

Dear KV,
I have spent the past three weeks trying to cherry-pick changes out of one branch into another. When do I just give up and merge?
In the Pits

PHOTOGRAPH BY ARINA HABICH

Dear Pits,
I once rode home with a friend from a computer conference in Monterey. It just so happened that this friend is a huge fan of fresh cherries, and when he saw a small stand selling baskets of them he stopped to buy some. Another trait this friend possesses is that he can't ever pass up a good deal. So while haggling with the cherry seller, it became obvious that buying a whole flat of cherries would be a better deal than buying a single basket, even though that was all we really wanted. Not wanting to pass up a deal, however, my friend bought the entire flat and off we went—eating and talking. It took another 45 minutes to get home, and during that time we had eaten more than half the flat of cherries. I could not look at anything even remotely cherry-flavored for months; and today, when someone says "cherry-picking," that does not conjure up happy images of privileged kids playing farmer on Saturday mornings along the California coast—I just feel ill.

All of which brings me to your letter. It is always difficult to say when someone else should "just give up and do X" no matter what X is, but at some point you know—deep down, somewhere in that place that makes you an engineer—what started out as a quick bit of cherry-picking has turned into a horrific slog through the mud, and nothing short of a John Deere tractor is going to get you out of it. The happy moments in the sunshine have ended, it is raining, you are cold, and you just want to go home. That is the time to stop and try again.

I know this probably ought to

go without saying, but the real reason most of us wind up in the pits of cherry-picking is because we have not been doing the real work of periodically merging whatever code we are working against. We have let the head of the tree, or the tip of the git, or whatever trite phrase people might want to use, get away from us, and the longer we wait to do the merge, the more pain we are going to suffer. The best way to keep from being stuck in the cherry orchard is to have a merged and tested branch ready to go when it is time for your project to resynchronize with the head of the development tree. I know this is more work than isolating yourself in a corner and just working on the next release, but in the end it will save you a lot of headaches. The question next time won't be, "When do I stop cherry-picking?" but simply, "When is the new branch ready to receive the work we have already done?"

KV

Dear KV,

I just started working for a new project lead who has an extremely annoying habit. Whenever I fix a bug and check in the fix to our code repository, she asks, "How do you know this is fixed?" or something like that, questioning every change I make to the system. It is as if she does not trust me to do my job. I always update our tests when I fix a bug, and that should be enough, don't you think? What does she want, a formal proof of correctness?

I Know Because I Know

Dear I Know,

Working on software is more than just knowing in your gut that the code is correct. In actuality, no part of working on software should be based on gut feelings, because, after all, software is supposed be a part of computer science, and science demands proof.

One of the problems I have with the current crop of bug-tracking systems—and trust me, this is only one of the problems I have with them—is that they do not do a good job tracking the work you have done to fix a bug. Most bug trackers have many states a bug can go through: new, open, analyzed, fixed, resolved, closed, and so forth, but that is only part of the story of fixing a bug, or doing anything else with a program of any size.

When you approach a problem, you should do it in a way that mirrors the scientific method.

A program is an expression of some sort of system that you, or a team, are implementing by writing it down as code. Because it is a system, one has to have some way of reasoning about that system. Many people will now leap up and yell, "Type Systems!", "Proofs!", and other things about which most working programmers have no idea and are not likely ever to come into contact with. There is, however, a simpler way of approaching this problem that does not depend on a fancy or esoteric programming language: use the scientific method.

When you approach a problem, you should do it in a way that mirrors the scientific method. You probably have an idea of what the problem is. Write that down as your theory. A theory explains some observable facts about the system. Based on your theory, you develop one or more hypothesis about the problem. A hypothesis is a *testable idea* for solving the problem. The nice thing about a hypothesis is that it is either true or false, which works well with our Boolean programmer brains: either/or, black or white, true or false.

The key here is to write all of this down. When I was young I never wrote things down because I thought I could keep them all in my head. But that was nonsense; I could not keep them all in my head, and I did not know the ones I had forgotten until my boss at the time asked me a question I could not answer. It is unsettling to realize you have a dumb look on your face in response to a question about something you are working on.

Eventually I developed a system of note taking that allowed me to make this a bit easier. When I have a theory about a problem, I create a note titled THEORY, and write down my idea. Under this, I write up all my tests (which I call TEST, because like any good programmer, I do not want to keep typing HYPOTHESIS). The note-taking system I currently use is Org mode in Emacs, which lets you create sequences that can be tied to hot keys, allowing you to change labels quickly. For bugs, I have labels called BUG, ANALYZED, PATCHED, |, and FIXED, while for hypotheses I have either PROVEN or DISPROVEN.

I always keep both the proven and disproven hypotheses. Why do I keep both? Because that way I know what I tried, and what worked and what failed. This proves to be invaluable when you have a boss with OCD, or, as they like to be called, "detail oriented." By keeping both your successes and failures, you can always go back, say in three months when the code breaks in a disturbingly similar way to the bug you closed, and look at what you tested last time. Maybe one of those hypotheses will prove to be useful, or maybe they will just remind you of the dumb things you tried, so you do not waste time trying them again. Whatever the case, you should store them, backed up, in some version-controlled way. Mine are in my personal source-code repository. You have your own repository, right? Right?!

KV

Related articles on queue.acm.org

Kode Vicious Bugs Out
George Neville-Neil
http://queue.acm.org/detail.cfm?id=1127862

Voyage in the Agile Memeplex
Philippe Kruchten
http://queue.acm.org/detail.cfm?id=1281893

ORM in Dynamic Languages
Chris Richardson
http://queue.acm.org/detail.cfm?id=1394140

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and a member 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.

DOI:10.1145/2492007.2492020

Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon

Education

Success in Introductory Programming: What Works?

How pair programming, peer instruction, and media computation have improved computer science education.

MANY *COMMUNICATIONS* READERS have been in faculty meetings where we have reviewed and bemoaned statistics about how bad attrition is in our introductory programming courses for computer science majors (CS1). Failure rates of 30%–50% are not uncommon worldwide.[1] There are usually as many suggestions for how to improve the course as there are faculty in the meeting. But do we know anything that really works?

We do, and we have research evidence to back it up. Pair programming, peer instruction, and media computation are three approaches to reforming CS1 that have shown positive, measurable impacts. Each of them is successful separately at improving retention or helping students learn, and combined, they have a dramatic effect.

Pair Programming

Pair programming is a practice that started in industry as an agile method. The idea is to have two people at a keyboard, one as the "driver" and the other as an "observer" or "navigator." The two people in the pair swap roles regularly while working. There is significant evidence that having two people at the keyboard improves productivity in industry, but does it help in the classroom?

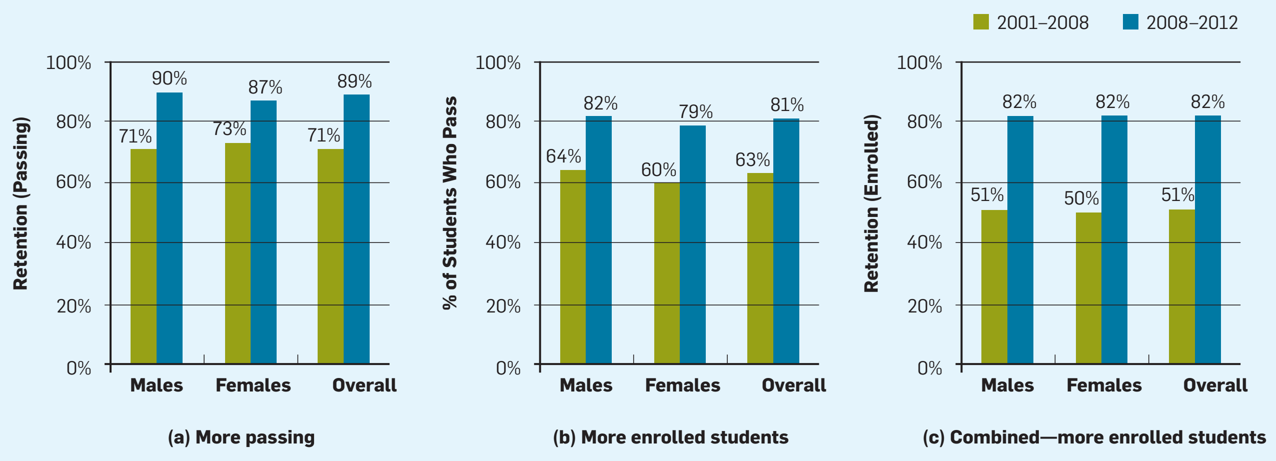
Pioneering work by Laurie Williams showed it could. Students using pair programming in an upper-division CS course produced higher-quality programs and learned the material faster. The idea is that students learn from the collaboration and the discussion, so the effort of coordinating work in a pair leads to better learning.

Charlie McDowell, Linda Werner, and their collaborators took this one step further. At the University of California at Santa Cruz (UCSC) they changed two sections of CS1 to use pair programming and left two sections with the usual solo work on programming assignments.[3] The researchers followed the students for one year after the first quarter course. They found that more students passed in the pairing sections (72%) versus the solo sections (63%), students were more likely to continue on into the next course (85% versus 67%), and they were more likely to have declared a CS major one year later (57% versus 34%). (Note: Many first-year students at UCSC are undeclared or have only a "proposed" major.)

Looking only at the final exam scores of the students that worked in pairs and

PHOTOGRAPH BY DIANA ARVAYO, COURTESY OF NEARSOFT INC.

Results of a course combining pair programming, peer instruction, and media computation over four years. (a) One-year retention for majors who pass introductory computing. (b) Passing rates for initially enrolled students. (c) One-year retention of initially enrolled students.



those that worked alone, there was no significant difference. It is important to note that significantly more students in the pairing section persisted to the end and took the final. This resulted in a higher percentage of students passing the course in the pairing sections. It also refutes the claim that weak students fail to learn the material because their partner does all of the work.

Media Computation

Georgia Tech requires all students to take a course in computer science, including students in Liberal Arts, Architecture, and Business majors. During the first four years of this requirement, the overall pass rate was 78%, which is quite reasonable. The pass rate for students in Liberal Arts, Architecture, and Business, however, was less than 50% on average.

Guzdial and his colleagues created a new course just for students in Liberal Arts, Architecture, and Business programs. For these students, computing is more about *communication* than *calculation*. Students in these programs most often use the computer in order to communicate with digital media.

Media Computation was an approach to CS1 that explained how digital media are manipulated. Students learned about loops by changing all the pixels in a picture to compute a negative image, or all the samples in a sound in order to decrease the volume. Students learned about conditionals by removing red eye in the image without changing any other colors, or changing only part of a sound.

What was most exciting about Media Computation was that our assignments were defined in terms of *computation*, but the choice of what media to use in the assignments was up to the students. Students produced beautiful and creative works of art—in their CS1 class.

The result on retention was pretty dramatic.[2] The pass rate for students in those majors went from below 50% in the former class to 85% in the Media Computation class. The research evidence in the computing education community suggests it is not just media. Giving students a *context* in which to apply and understand computing makes it more relevant, and makes the students more successful.

We believe each of these approaches addresses a failing of traditional introductory computing courses.

Peer Instruction

Many of us have had the experience lecturing to a class where we have explained a key concept with brilliant clarity. We turn to the class, ask if there are any questions, and we hear... crickets. One-half of the class is looking at phones or laptops, one-fourth looks utterly confused and scared, and another one-fourth looks bored. No one asks anything, you think "they've got it" and move on. After the exam you discover: they didn't get it.

Peer Instruction, originally developed by Eric Mazur for teaching physics, seeks to remedy this problem by engaging students in the learning process. Peer Instruction modifies the standard "lecture" to revolve around 3–5 questions per lecture. For each of these questions, students follow the PI process: individually think about the problem and answer (often using clickers); discuss the question in groups; then answer again (using a clicker). Lastly, the instructor leads a class-wide discussion on the question and dynamically adjusts their explanation based on student performance.

Peer Instruction in physics has consistently shown twofold improvements in student performance on concept inventory exams versus standard lecture in large multi-institutional studies. Although Peer Instruction is new to computing, computer science education research has shown that students value Peer Instruction in upper and lower division classes, instructors value Peer Instruction, students learn from peer discussion, students in Peer Instruction classes experience a 61% reduction in failure rates, and students in Peer Instruction classes outperform standard lecture by 5% on identical final exams. (All references can be found at http://www.peerinstruction4cs.org/latest-research/.)

Combining All Three at UCSD

At this year's SIGCSE Symposium, Porter and Simon[5] reported on how all three of these approaches were combined in an introductory programming course at University of California at San Diego (UCSD). They started tracking students in 2001, and in 2008, created a new quarter-long course that combined pair programming, peer instruction, and media computation. After running the new course for four years, the results were remarkable. Not only were more students *who passed the class* retained into the Sophomore year (section a in the figure), but because more students also passed among those who initially enrolled (section b in the figure) the combined effect had a dramatic impact on retention of students enrolling in CS1 (section c in the figure).

Why Did More Students Succeed?

What is going on in these three reform efforts that cause this large change in retention? We believe each of these approaches addresses a failing of traditional introductory computing courses. We hear an often-repeated set of complaints about computer science education:

- *Computer science is asocial. Students see it being about sitting in the corner and hacking for hours on end, and that's just not attractive.* Pair programming shows students that being in computer science is about an intense social experience, and that learning and performance in computer science is made better by working with others.
- *Computer science is tedious, boring, and irrelevant.* Media Computation shows students that computer science is a creative endeavor, where the output can be beautiful. At UCSD and Skidmore College, these classes have begun hosting campuswide art shows to showcase student work,[4] a far cry from being asocial and irrelevant.
- *Computer science classes are competitive, with students focused on their individual grade.* Peer instruction shows students that computer science lectures are about collaborating to learn and working together as a team—starting preparation for effective work in software development teams.

There is a natural response to these kinds of efforts: that we just made these courses "easier" or "dumbed them down." The data we present about *greater* success rates into the second year, after changing only a single course, suggests students are at least as well prepared after implementing these reforms. As long as the students are achieving *desired course outcomes*, we *should* aim to make the class easier. There is no great virtue in a difficult course that flunks out students. These results demonstrate that research-based practices can make a course "easier," with higher pass rates and higher long-term retention, while still achieving desired learning outcomes. C

References

1. Bennedsen, J. and Caspersen, M.E. Failure rates in introductory programming. *SIGCSE Bull. 39*, 2 (2007), 32–36.
2. Guzdial, M. and Elliott Tew, A. Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education. In *Proceedings of the Second International Workshop on Computing Education Research*. (2006).
3. McDowell, C., Werner, L., Bullock, H.E., and Fernald, J. Pair programming improves student retention, confidence, and program quality. *Commun. ACM 49*, 8 (Aug. 2006), 90–95; DOI: 10.1145/1145287.1145293.
4. Porter, L. and Simon, B. Fostering creativity in CS1 by hosting a computer science art show. *ACM Inroads* (Mar. 2013).
5. Porter, L. and Simon, B. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceedings of the 44th Special Interest Group on Computer Science Education Technical Symposium* (2013).

Leo Porter (lporter1@skidmore.edu) is an assistant professor of computer science at Skidmore College, Saratoga Springs, NY.

Mark Guzdial (guzdial@cc.gatech.edu) is a professor in the College of Computing at Georgia Institute of Technology in Atlanta, GA.

Charlie McDowell (charlie@soe.ucsc.edu) is a professor of computer science and Associate Dean for Undergraduate Affairs in the Basking School of Engineering at the University of California Santa Cruz.

Beth Simon (bsimon@cs.ucsd.edu) is a lecturer in computer science and engineering and the director of the Center for Teaching Development at the University of California, San Diego.

DOI:10.1145/2492007.2492021

Jean-Loup Richet

Viewpoint

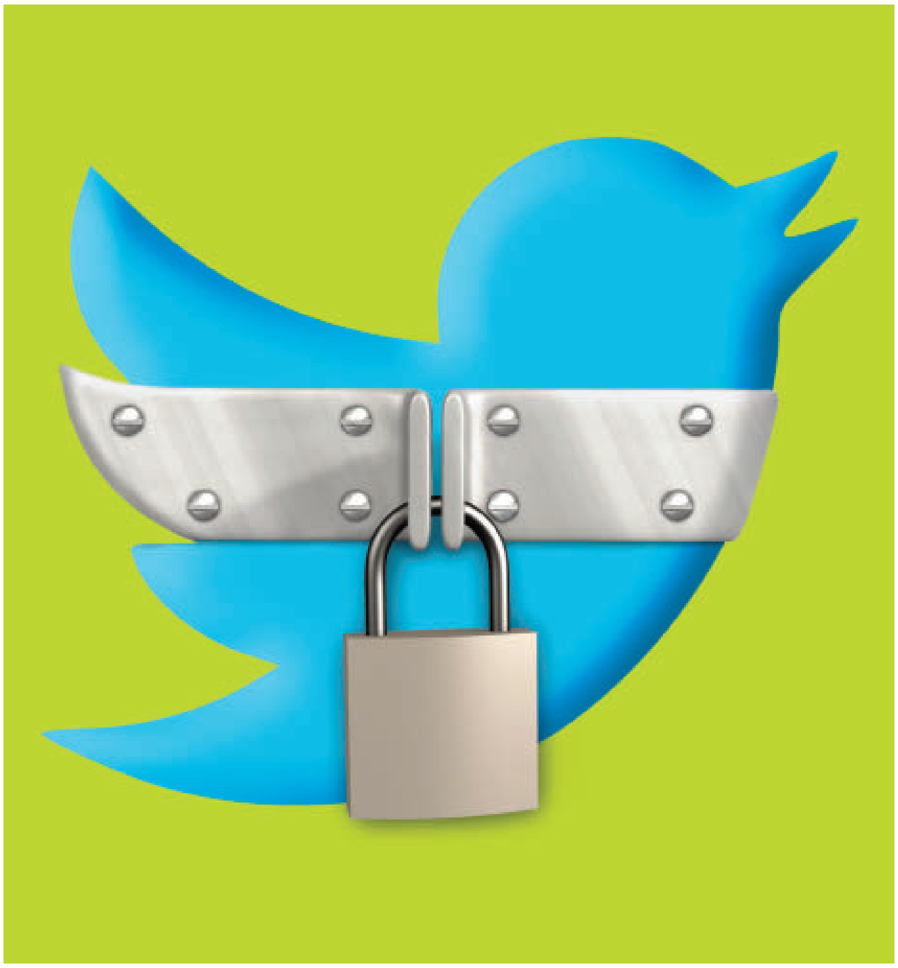
Overt Censorship: A Fatal Mistake?

Censorship of information often has the opposite effect by drawing attention to the censored material.

Recently, Twitter has reignited debate over Internet censorship by making a German neo-Nazi account unavailable to users in Germany at the request of German authorities.[2] This followed Twitter's adoption of a "country withheld content" policy in January 2012, which allows Twitter to block content in certain countries upon government request.[a] Since adopting the policy, Twitter has reportedly received several government requests to make sites unavailable in accordance with this policy, but for undisclosed reasons, Twitter has failed to act on any of the requests.

In Germany, the use of Nazi symbols is strictly forbidden by law, as is membership in a neo-Nazi organization. The clear illegality of the material in Germany is perhaps what made Twitter's decision in this case so easy to make. All of the material is still available outside of Germany, however, and users have claimed it is relatively easy to bypass the blockage (using proxy servers, VPNs, and other methods).

Meanwhile, China's government has recently been criticized for its decision to block access to a *New York Times* article[1] that investigated assets accumulated by its prime minister, Wen Jiabao. China's censors were nothing if not thorough, even deleting all social media posts that made



reference to the issue and a Sina Weibo (a microblogging platform similar to Twitter) account that promoted the Chinese language version of the *New York Times*' arts and culture section. Despite this attention to detail, however, many thousands of users in China continued to discuss the issue by making veiled references to it, and by using deliberate misspellings and other covert tactics.

Interestingly, Twitter's blockage of the neo-Nazi Better Hannover (@hannoverticker) stimulated interest

a Twitter's Terms of Service: https://support.twitter.com/articles/20169222#.

in the group, causing its number of followers to grow rapidly by 200 in just one day. Whether those followers were neo-Nazi sympathizers or simply people who were interested in the case is unclear, but what is clear is that the censorship itself acted as global publicity. It is arguable that in both cases the furor caused by the censorship of information has only served to draw attention to it further, surely defeating its purpose.

In fact, ever since the samizdat[b] of the Soviet bloc, it has been clear that in many ways overt censorship has always served to stimulate interest in the forbidden material. In the case of samizdat, this forbidden material became strongly fetishized in such a way as to make its contents immediately appealing, and immediately accepted by many sections of the population as truth. As Ann Komaromi has said in her essay "The Material Existence of Soviet Samizdat," "[Samizdat was]… something on which to get high… … an intoxicating product. It was forbidden fruit."[4] The process of overt censorship here only served to elevate the censored material to a sacred status. It is clear that it also serves to expose a government's intentions, and in many cases to undermine the government's credibility.

Why is it, then, that governments such as China's still choose to pursue a policy of aggressive and overt Internet censorship? The issue was investigated by researchers at Harvard University.[3] The conclusion they came to was that China's policy on censorship is not what it seems. From the outside, it may seem China's intention is to suppress all anti-government discussion and the expression of revolutionary ideas. In fact, what China found was that direct criticism of the government was no more likely to be censored than anything else. Censorship was instead targeted at postings that sought specifically to organize collective anti-government action or to create action groups of any kind.

Germany's reasoning for the censorship of Better Hannover was similar. Germany's aim was not the suppression of neo-Nazi ideas, but rather the prevention of any organized and collective anti-democratic action suggested by those ideas.

Censorship exposes a government's intentions and in many cases undermines the government's credibility.

It is easy to see why Germany would adopt such a policy. After the horrors that National Socialism inflicted on the German population, it was clear that measures had to be taken to stop such a situation from arising in the future. Unfortunately, however, these measures may only serve to stimulate interest in the subject, and to create a whole new kind of samizdat in the form of websites that must be accessed covertly. They may be successful in disabling action by these forces for a certain period of time, but it remains to be seen in both cases whether the publicity created by overt and aggressive censorship ultimately strengthens or weakens the government's position. C

b 'Self-editing and publishing' in times of censorship. For more details, see http://bit.ly/183hLUU.

References

1. Bradsher, K. China criticizes the Times for article on premier's family fortune. *The New York Times* (Oct. 26, 2012); http://www.nytimes.com/2012/10/27/world/asia/china-criticizes-the-times-for-article-on-premiers-family-fortune.html?ref=censorship.
2. Brumfield, B. and Smith-Spark, L. Twitter blocks content of German neo-Nazi group. CNN (Oct. 10, 2012); http://articles.cnn.com/2012-10-18/tech/tech_twitter-censorship_1_alex-macgillivray-twitter-neo-nazi.
3. King, G., Pan, J., and Roberts, M. How censorship in China allows government criticism but silences collective expression; http://gking.harvard.edu/gking/files/censored.pdf.
4. Komaromi, A. The material existence of Soviet Samizdat. *Slavic Review 63*, 3 (Autumn 2004), 605.

Jean-Loup Richet (jeanloup.richet@gmail.com) is an information systems service manager at Orange in Paris, France, and Ph.D. student at Nantes University (LEMNA), Nantes, France.

ACM's *Career & Job Center!*

Are you looking for your next IT job?

Do you need Career Advice?

Visit ACM's Career & Job Center at:

http://www.acm.org/careercenter

◆ ◆ ◆ ◆ ◆

The **ACM Career & Job Center** offers ACM members a host of career-enhancing benefits:

- A highly targeted focus on job opportunities in the computing industry
- Access to hundreds of corporate job postings
- Resume posting keeping you connected to the employment market while letting you maintain full control over your confidential information
- An advanced Job Alert system that notifies you of new opportunities matching your criteria
- Career coaching and guidance from trained experts dedicated to your success
- A content library of the best career articles compiled from hundreds of sources, and much more!

Association for Computing Machinery

Advancing Computing as a Science & Profession

The **ACM Career & Job Center** is the perfect place to begin searching for your next employment opportunity!

Visit today at

http://www.acm.org/careercenter

DOI:10.1145/2492007.2492022

Article development led by acmqueue
queue.acm.org

Embracing failure to improve resilience and maximize availability.

BY ARIEL TSEITLIN

The Antifragile Organization

FAILURE IS INEVITABLE. Disks fail. Software bugs lay dormant waiting for just the right conditions to bite. People make mistakes. Data centers are built on farms of unreliable commodity hardware. If you are running in a cloud environment, then many of these factors are outside of your control. To compound the problem, failure is not predictable and does not occur with uniform probability and frequency. The lack of a uniform frequency increases uncertainty and risk in the system. In the face of such inevitable and unpredictable failure, how can you build a reliable service that provides the high level of availability your users can depend on?

A naive approach could attempt to prove the correctness of a system through rigorous analysis. It could model all different types of failures and deduce the proper workings of the system through a simulation or another theoretical framework that emulates or analyzes the real operating environment. Unfortunately, the state of the art of static analysis and testing in the industry has not reached those capabilities.[4]

A different approach could attempt to create exhaustive test suites to simulate all failure modes in a separate test environment. The goal of each test suite would be to maintain the proper functioning of each component, as well as the entire system when individual components fail. Most software systems use this approach in one form or another, with a combination of unit and integration tests. More advanced usage includes measuring the coverage surface of tests to indicate completeness.

While this approach does improve the quality of the system and can prevent a large class of failures, it is insufficient in maintaining resilience in a large-scale distributed system. A distributed system must address the challenges posed by data and information flow. The complexity of designing and executing tests that properly capture the behavior of the target system is greater than that of building the system itself. Layer on top of that the attribute of large scale, and it becomes unfeasible, with current means, to achieve this in practice while maintaining a high velocity of innovation and feature delivery.

Yet another approach, advocated in this article, is to induce failures in the system to empirically demonstrate resilience and validate intended behavior. Given the system was designed with resilience to failures, inducing those failures—within original design parameters—validates the system behaves as expected. Because this approach uses the actual live system, any resilience gaps that emerge are identified and caught quickly as the system evolves and changes. In the second approach just described, many complex issues are not caught in the test environment and manifest themselves in unique and infrequent ways only in the live environment. This, in turn, increases the likelihood of latent bugs remaining undiscovered and accumu-

IMAGE FROM SHUTTERSTOCK.COM

lating, only to cause larger problems when the right failure mode occurs. With failure induction, the added need to model changes in the data, information flow, and deployment architecture in a test environment is minimized and presents less of an opportunity to miss problems.

Before going further, let's discuss what is meant by resilience and how to increase it.

Resilience is an attribute of a system that enables it to deal with failure in a way that does not cause the entire system to fail. It could involve minimizing the blast radius when a failure occurs or changing the user experience to work around a failing component. For example, if a movie recommendation service fails, the user can be presented with a nonpersonalized list of popular titles. A complex system is constantly undergoing varying degrees of failure. Resiliency is the measure by which it can recover, or be insulated, from failure, both current and future.[7]

There are two ways of increasing the resilience of a system:

- *Build your application with redundancy and fault tolerance.* In a service-oriented architecture, components are encapsulated in services. Services are made up of redundant execution units (instances) that protect clients from single- or multiple-unit failure. When an entire service fails, clients of that service must implement fault tolerance to localize the failure and continue to function.
- *Reduce uncertainty by regularly inducing failure.* Increasing the frequency of failure reduces its uncertainty and the likelihood of an inappropriate or unexpected response. Each unique failure can be induced while observing the application. For each undesirable response to an induced failure, the first approach can be applied to prevent its recurrence. Although in practice it is not feasible to induce every possible failure, the exercise of enumerating possible failures and prioritizing them helps in understanding tolerable operating conditions and classifying failures when they fall outside those bounds.

The first item is well covered in other literature. The remainder of this article will focus on the second.

The Simian Army

Once you have accepted the idea of inducing failure regularly, there are a few choices on how to proceed. One option is GameDays,[1] a set of scheduled exercises where failure is manually introduced or simulated to mirror real-world failure with the goal of both identifying the results and practicing the response—a fire drill of sorts. Used by the likes of Amazon and Google, GameDays are a great way to induce failure on a regular basis, validate assumptions about system behavior, and improve organizational response.

But what if you want a solution that is more scalable and automated—one that does not run once per quarter but rather once per week or even per day? You do not want failure to be a fire drill. You want it to be a nonevent—something that happens all the time in the background so that when a real failure occurs, it will simply blend in without any impact.

One way of achieving this is to engineer failure to occur in the live environment. This is how the idea for "monkeys" (autonomous agents really, but monkeys inspire the imagination) came to Netflix to wreak havoc and induce failure. Later the monkeys were assembled together and labeled the Simian Army.[5] A description of each resilience-related monkey follows.

Chaos Monkey. The failure of a virtual instance is the most common type of failure encountered in a typical public cloud environment. It can be caused by a power outage in the hosting rack, a disk failure, or a network partition that cuts off access. Regardless of the cause, the result is the same: the instance becomes unavailable. Inducing such failures helps ensure services do not rely on any on-instance state, instance affinity, or persistent connections.

To address this need, Netflix created its first monkey: Chaos Monkey, which randomly terminates virtual instances in a production environment—instances that are serving live customer traffic.[3]

Chaos Monkey starts by looking into a service registry to find all the services that are running. In Netflix's case, this is done through a combination of Asgard[6] and Edda.[2] Each service can override the default Chaos Monkey configuration to change termination probability or opt out entirely. Each hour, Chaos Monkey wakes up, rolls the dice, and terminates the affected instances using Amazon Web Services (AWS) APIs.

Chaos Monkey can optionally send an email message to the service owner when a termination is made, but most service owners do not enable this option because instance terminations are common enough occurrences that they do not cause any service degradation.

Chaos Gorilla. With Chaos Monkey, a system is resilient to individual instance failure, but what if an entire data center was to become unavailable? What would be the impact to users if an entire Amazon availability zone (AZ) went offline? To answer that question and to make sure such an event would have minimal customer impact, Netflix created Chaos Gorilla.

Chaos Gorilla causes an entire AZ to fail. It simulates two failure modes:

- *Network partition.* The instances in the zone are still running and can communicate with each other but are unable to communicate with any service outside the zone and are not reachable by any other service outside the zone.
- *Total zone failure.* All instances in the zone are terminated.

Chaos Gorilla causes massive damage and requires a sophisticated control system to rebalance load. For Net flix, that system is still being developed, and as a result, Chaos Gorilla is run manually, similar to the GameDay exercises mentioned previously. With each successive run, Chaos Gorilla becomes more aggressive in the way it executes the failures—the goal being to run it in an automated unattended way as in Chaos Monkey.

Chaos Kong. A region is made up of multiple data centers (availability zones) that are meant to be isolated from one another. A robust deployment architecture has AZ redundancy by using multiple AZs. In practice, regionwide failures do occur, which makes single-region deployments insufficient in providing resilience to regionwide failures. Once a system is deployed redundantly to multiple regions, region failure must be tested analogously to instances and availability zones. Chaos Kong serves that purpose. Netflix is working toward the goal of taking an entire region offline with Chaos Kong.

Latency Monkey. Once Chaos Monkey is running and individual instance failure no longer has any impact, a new class of failures emerges. Dealing with instance failure is relatively easy: just terminate the bad instances and let new healthy instances take their places. Detecting when instances become unhealthy, but are still working, is more difficult, and having resilience to this failure mode is harder still. Error rates could become elevated, but the service could occasionally return success. The service could reply with successful responses, but latency could increase, causing timeouts.

What Netflix needed was a way of inducing failure that simulated partially healthy instances. Hence came the genesis of Latency Monkey, which induces artificial delays in the RESTful client-server communication layer to simulate service degradation and measures if upstream services respond appropriately. In addition, by creating very large delays, node downtime, or even an entire service downtime, can be simulated without physically bringing instances or services down. This can be particularly useful when testing the fault tolerance of a new service by simulating the failure of its dependencies, without making these dependencies unavailable to the rest of the system.

The remaining army. The rest of the Simian Army, including Janitor Monkey, takes care of upkeep and other miscellaneous tasks not directly related to availability. (For details, see http://techblog.netflix.com/2011/07/netflix-simian-army.html.)

Monkey Training at Netflix

While the Simian Army is a novel concept and may require a shift in perspective, it is not as difficult to implement as it initially appears. Understanding what Netflix went through is illustrative for others interested in following such a path.

Netflix is known for being bold in its rapid pursuit of innovation and high availability, but not to the point of callousness. It is careful to avoid any noticeable impact to customers from these failure-induction exercises. To minimize risk, Netflix takes the following steps when introducing a monkey:

1. With the new monkey in the test environment, engineers observe the user experience. The goal is to have negligible or zero impact on the customer. If the engineers see any adverse results, then they make the necessary code changes to prevent recurrence. This step is repeated as many times as necessary until no adverse user experience is observed.
2. Once no adverse results are observed in the test environment, the new monkey is enabled in the production environment. Initially, the new monkey is run in opt-in mode. One or more services are selected to run the new monkey against, having already been run in the test environment. The new monkey runs for a few months in this mode, opting in new services over time.
3. After many services have opted in, the new monkey graduates to opt-out mode, in which all services are potential targets for the new monkey. If a service is placed in an opt-out list, the monkey avoids it.
4. The opt-out list is periodically reviewed for each monkey, and service owners are encouraged to remove their services from the list. The platform and monkey are improved to increase adoption and address reasons for opting out.

A complex system is constantly undergoing varying degrees of failure. Resiliency is the measure by which it can recover, or be insulated, from failure, both current and future.

The Importance of Observability

No discussion of resilience would be complete without highlighting the important role of monitoring. *Monitoring* here means the ability to observe and, optionally, signal an alarm on the external and internal states of the system and its components. In the context of failure induction and resilience, monitoring is important for two reasons:

- During a real, nonsimulated customer-impacting event, it is important to stabilize the system and eliminate customer impact as quickly as possible. Any automation that causes additional failure must be stopped during this time. Failing to do so can cause Chaos Monkey, Latency Monkey, and the other simians to further weaken an already unhealthy system, causing even greater adverse end-user impact. The ability to observe and detect customer-impacting service degradation is an important prerequisite to building and enabling automation that causes failure.

▸ Building resilient systems does not happen at a single point in time; it is an ongoing process that involves discovering weaknesses and dealing with them in an iterative learning cycle. Deep visibility into the system is key to understanding how the system operates and in which ways it fails. Few root-cause investigations would succeed without metrics and insights into operations of the system and its components. Monitoring provides a deep understanding of how the system operates, especially when it fails, and makes it possible to discover weaknesses in the system and identify anti-patterns for resilience.

One of the most important first questions to ask during a customer-impacting event is, "What changed?" Therefore, another key aspect of monitoring and observability is the ability to record changes to the state of the system. Whether a new code deployment, a change in runtime configuration, or a state change by an externally used service, the change must be recorded for easy retrieval later. Netflix built a system, internally called Chronos, for this purpose. Any event that changes the state of the system is recorded in Chronos and can be quickly queried to aid in causality attribution.

The Antifragile Organization

Resilience to failure is a lofty goal. It enables a system to survive and withstand failure. There is an even higher peak to strive for, however: making the system stronger and better with each failure. In Nassim Taleb's parlance, it can become *antifragile*—growing stronger from each successive stressor, disturbance, and failure.[8]

Netflix has taken the following steps to create a more antifragile system and organization:

1. *Every engineer is an operator of the service.* This is sometimes referred to in jest as "no ops," though it is really more "distributed ops." Separating development and operations creates a division of responsibilities that can lead to a number of challenges, including network externalities and misaligned incentives. Network externalities are caused by operators feeling the pain of problems that developers introduce. Misaligned incentives are a result of operators wanting stability while developers desire velocity. The DevOps movement was started in response to this divide. Instead of separating development and operations, developers should operate their own services. They deploy their code to production and then they are the ones awakened in the middle of the night if any part of it breaks and impacts customers. By combining development and operations, each engineer can respond to failure by altering the service to be more resilient to and fault tolerant of future failures.

2. *Each failure is an opportunity to learn, generating these questions:* "How could the failure have been detected more quickly?" "How can the system be more resilient to this type of failure?" "How can this failure be induced on a regular basis?" The result is each failure makes the system more robust and resilient, analogous to the experience a warrior gains in each battle to make him stronger and fiercer in the next. The system becomes better the more times and ways it fails.

3. *A blameless culture is fostered.* As an organization, Netflix optimizes for innovation and velocity, and it accepts that mistakes will sometimes occur, using each one as an opportunity to learn. A commonly overheard saying at Netflix is, "If we're not making any mistakes, it means we're not moving quickly enough." Mistakes are not a bad thing, unless the same mistakes are made over and over again. The result is that people are less worried about making mistakes, and postmortems can be structured as effective opportunities to learn (see step 2).

Conclusion

The more frequently failure occurs, the more prepared the system and organization become to deal with it in a transparent and predictable manner. Inducing failure is the best way of ensuring both system and organizational resilience. The goal is to maximize availability, insulating users of a service from failure and delivering a consistent and available user experience. Resilience can be improved by increasing the frequency and variety of failure and evolving the system to deal better with each newfound failure, thereby increasing antifragility. Focusing on learning and fostering a blameless culture are essential organizational elements in creating proper feedback in the system. C

Related articles on queue.acm.org

Automating Software Failure Reporting
Brendan Murphy
http://queue.acm.org/detail.cfm?id=1036498

Keeping Bits Safe: How Hard Can It Be?
David S. H. Rosenthal
http://queue.acm.org/detail.cfm?id=1866298

Monitoring, at Your Service
Bill Hoffman
http://queue.acm.org/detail.cfm?id=1113335

References

1. ACM. Resilience engineering: Learning to embrace failure. *Commun. ACM 55*, 11 (Nov. 2012), 40–47; http://dx.doi.org/10.1145/2366316.2366331.
2. Bennett, C. Edda—Learn the stories of your cloud deployments. The Netflix Tech Blog; http://techblog.netflix.com/2012/11/edda-learn-stories-of-your-cloud.html.
3. Bennett, C. and Tseitlin, A. Chaos Monkey released into the wild. The Netflix Tech Blog; http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html.
4. Chandra, T.D., Griesemer, R. and Redstone, J. Paxos made live: An engineering perspective. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing* (2007), 398–407; http://labs.google.com/papers/paxos_made_live.pdf.
5. Izrailevsky, Y. and Tseitlin, A. The Netflix Simian Army. The Netflix Tech Blog; http://techblog.netflix.com/2011/07/netflix-simian-army.html.
6. Sondow, J. Asgard: Web-based cloud management and deployment. The Netflix Tech Blog; http://techblog.netflix.com/2012/06/asgard-web-based-cloud-management-and.html.
7. Strigini, L. Fault tolerance and resilience: meanings, measures and assessment. Centre for Software Reliability, City University, London, U.K., 2009; http://www.csr.city.ac.uk/projects/amber/resilienceFTmeasurementv06.pdf.
8. Taleb, N. *Antifragile: Things That Gain from Disorder.* Random House, 2012.

Ariel Tseitlin is director of cloud solutions at Netflix where he manages the Netflix Cloud and is responsible for cloud tooling, monitoring, performance and scalability, and cloud operations and reliability engineering. He is also interested in resilience and highly available distributed systems. Prior to joining Netflix, he was most recently VP of technology and products at Sungevity and before that was the founder and CEO of CTOWorks.

DOI:10.1145/2492007.2492023

Article development led by acmqueue
queue.acm.org

Which practices should be modified or avoided altogether by developers for the mobile Web?

BY ALEX NICOLAOU

Best Practices on the Move: Building Web Apps for Mobile Devices

IF IT WAS not your priority last year or the year before, it is sure to be your priority now: bring your website or service to mobile devices in 2013 or suffer the consequences. Early adopters have been talking about mobile taking over since 1999—anticipating the trend by only a decade or so. Today, mobile Web traffic is dramatically on the rise, and creating a slick mobile experience is at the top of everyone's mind. Total mobile data traffic is expected to exceed 10 exabytes per month by 2017, as shown in Figure 1 (in case your mind is not used to working in exabytes yet, that is 10 million terabytes per month, or almost four terabytes per second).[1]

Of that data, iOS and Android devices consume the lion's share, which suggests a focus for any immediate Web development efforts. Figure 2 shows a breakout of megabytes per month downloaded in 2011 and 2012, indicating these platforms are widely used and trending upward.

So, piece of cake, right? Just take some of the great literature on writing desktop Web sites and apply it to mobile. For example, Yahoo!'s excellent YSlow tool and corresponding performance rules[8] are an excellent starting point; Google's PageSpeed provides similar guidelines.[5]

The best practices for optimizing website performance, as articulated in YSlow and PageSpeed, were developed largely with the desktop world in mind. Now that mobile devices need to be accounted for as well, some of those practices might not have the intended benefits—or might even degrade performance on mobile.

Mobile users expect a more application-like experience. Smooth load experiences, fast and animated transitions, and application-centric error messages are part of what makes using Internet services on mobile devices bearable and perhaps even enjoyable in spite of slow networks, tiny screens, and cold fingers (at least for those of us in the north). Mobile users demand that you deliver more with less, and the old rules of website design and implementation for the desktop need a lot of adaptation to create a slick mobile Web-app experience.

Which Recommendations Should Be Followed?

Although the recommendations in Google's PageSpeed and Yahoo!'s YSlow work well for desktop development, some of them need to be reconsidered to provide maximum benefit for mobile development. Network connectivity, while far more ubiquitous than ever before, is still spotty; a mobile site needs to compensate for this problem so users generally feel the app is responsive even when the network is not. Older phones parse and execute JavaScript 100 times slower than desktops, and even the latest phones are still slower than desktop devices by a factor of 10—and this slowdown magnifies the smallest of sins. Handling JavaScript and network problems well will bring your performance within the expectations for any native mobile app. Then all that will remain is a host of fit-and-finish problems that require a solid understanding of when and why the browser paints and lays out to add the final polish.

With these thoughts in mind, let's consider how YSlow and PageSpeed guidelines apply to the world of Android and iPhone Web-app development.

Eliminate HTTP Requests and Round Trips. A core recommendation from both Google and Yahoo! is to minimize HTTP requests. This is a critical concept that needs special expertise and care to be effective on mobile. The good news is that existing recommendations on using CSS sprites to represent images embedded as inline `data:URLs` are techniques that work well on mobile.

Image maps should work as well, but their inflexibility in terms of layout options is a bit of a drawback for mobile. For example, for a truly slick mobile experience you will want to make good use of orientation changes that affect the horizontal dimension of your Web page. With image maps you would either have to use two of them or dice them up, in which case you might as well use CSS sprites and do your layout with markup as usual. So while image maps should work as a technique, on the whole sprites work better.

These existing techniques alone will not be enough to supercharge page load on mobile. On iOS, for example, pages are cached in memory only, and HTML files still appear to be limited to 25KB uncompressed (though other resource types can be much larger now). In addition, some iOS devices are still limited to a maxi-

Figure 1. Mobile data traffic projection.

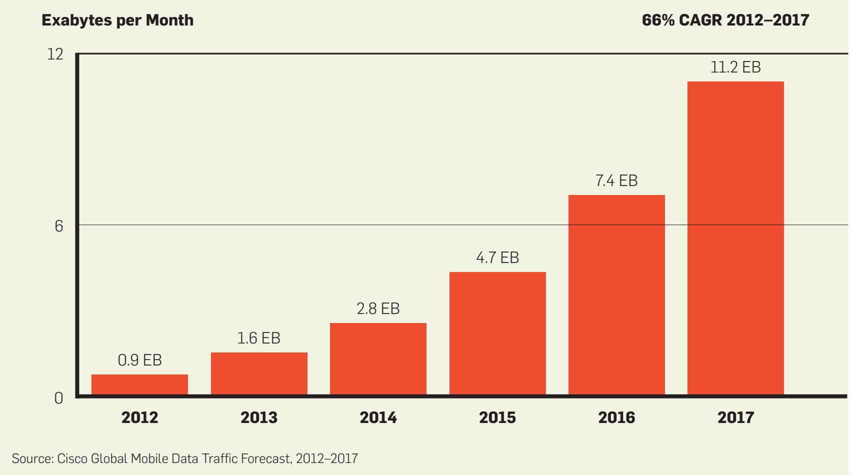
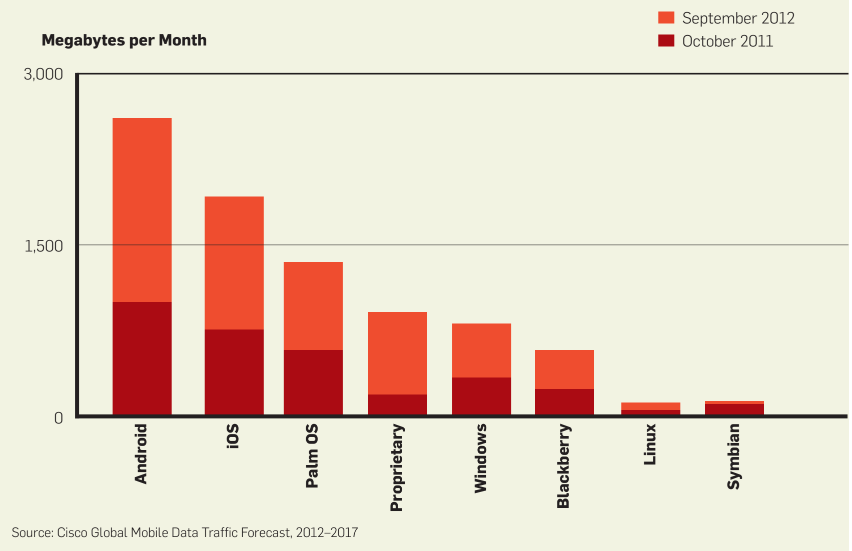


Figure 2. Megabytes per month by year and platform. Figure 2. Megabytes per month by year and platform.



mum of four parallel connections to download your content—which means you cannot afford to rely on multiple external sources to load in a reasonable amount of time. Finally, the HTTP 1.1 specification recommends at most two parallel downloads per hostname. Of course these values are changing all the time; to find tools and data on the latest limits, Steve Souder's Mobile Cache File Sizes[7] and Ryan Grove's Mobile Browser Cache Limits, Revisited[6] are good starting points.

If you truly want to solve the cache issue, it is important to force the mobile browser to cache all the unchanging content on a site permanently, using HTML5 application cache, commonly referred to as app cache. Because app cache is still on the bleeding edge, it is best to use the minimal subset possible for your application.

The cache is controlled by a single manifest file that tells the browser where to get resources. The minimum functionality would be to make your website load as a single large Web page, and have the manifest file list that page and a hashCode as a comment that is updated when the Web page changes. Dynamic content can be loaded via XHR (`XMLHttpRequest`) and inserted into the document. This technique is most effective if you design the prefix of your page (up to about the 1,400th byte, in order to fit into a single packet across almost any TCP/IP network) to render a basic framework for the page before any script is executed.

If you design your site in this way, you can also have the framework of the page fade in by providing an opacity transition from 0.0001 to 1.0 just after the framework of markup is loaded. This will create a very slick "app-like" startup experience. The opacity property is the method of choice for smooth transitions because the browser will have prerendered everything that is at opacity 0.0001, but the user cannot see it, and fading in is slick and smooth. Using opacity 0 will cause a white flash when the browser repaints, and using the display CSS attribute will be even worse because it will cause re-layout. If you cannot get the basic framework of your page to fit into the first packet, then you can instead load a spinner or a logo, again at opacity 0.0001, and conditionally detect whether the app cache is already populated before showing the spinner. The JavaScript for accomplishing this should appear in a script tag after the markup, as illustrated in Figure 3.

Figure 3. Reduce abandonment with a conditional spinner for non-approached loads.

```
<script>
if (window.applicationCache.status == 0) {
 // Page was loaded from the Network; reveal the spinner
} else {
 // Page was loaded from AppCache; heavier-weight startup applies
}
</script>
```

Using this pattern ensures the user will immediately get the feedback that your page is loading, thus reducing the chance of abandonment at the most critical moment: when the user first discovers your site.

To complete the effect, you also need to ensure your serving infrastructure flushes the framework of the page to the network before doing any heavy lifting on the server side, so that all overhead on first request is deferred until after the initial packet is sent. This will prove remarkably speedy even on spottier connections. It is also not a bad idea to make a request to the server at this point to track the page-load metric (more on this later in the section on JavaScript parse performance).

Following the recommended pattern means that even on the first load, all code, assets, and CSS for your site will load in a single round trip, and that will create the best possible mobile experience. Dynamic data will of course have to be loaded separately, and the same level of care and caching should apply there, too, except that HTML5 storage APIs will have to be used to cache follow-on content. Fewer than four round trips—one for the DNS lookup, one for the initial page content, and up to two for dynamic content—should produce a responsive mobile experience for your users.

Page Speed's best practices for minimizing round-trip times[4] also include a number of techniques for avoiding round trips you might overlook, such as DNS lookups and redirects. Almost every recommendation relating to round-trip times is well worth implementing, as Google has provided good techniques for combining the content that forms all the components of your site.

The last item in that section of Google's recommendations focuses on parallelism achieved by serving from multiple hostnames; but parallel downloads are likely to prove ineffective for mobile devices. Combining your resources and components into single files and ensuring they are served from app cache will be the most effective technique.

Use Compression. Using compression on all content that would benefit (essentially everything except images and video) remains a great recommendation for mobile. Mobile CPUs are getting faster much more quickly than mobile networks, and their speed does not change when a user is visiting the cottage. Even if you have diligently used app cache and local storage to save all resources locally on the device, you can expect the user will be regularly loading dynamic content, and all these requests should be compressed. To speed initial load time, even cached content should be delivered compressed whenever possible, though the benefit is less.

Manage JavaScript Parse Time. After caching and compression, script loading is likely to be the single biggest source of performance degradation for a website, and it is the more difficult to address. The big issue on mobile is that parse and execute times of script are much slower than one would expect. In fact, the rule of thumb is that parse and execute times are 10 times slower than when testing on a desktop. A JavaScript payload of as little as 100KB can cost 100ms of startup time even on reasonably recent phones such as iPhone 4—and the previous generation of iPhone was 10 times slower!

Therefore, it is certainly worth putting the vast bulk of script at the bottom of the page. To evaluate parse time, you can use two script tags as shown in Figure 4.

The start checkpoint is parsed and evaluated before parsing for the sec-

Figure 4. Monitor the time required to parse and evaluate your JavaScript.

```
<script>
var start_checkpoint = new Date();
</script>
<script>
var parsed_checkpoint = new Date();
// more script here
var eval_checkpoint = new Date();
var parse_time = parse_checkpoint - start_checkpoint;
var eval_time = eval_checkpoint - parse_checkpoint;
</script>
```

Figure 5. Declaring a block of JavaScript for deferred parsing.

You can compute the load time of the page with:

```
 window.performance.responseEnd  -  window.performance.navigationStart
```

and document load time with:

```
window.performance.loadEventEnd - window.performance.responseEnd.
```

ond script tag begins. The second script is then loaded and parsed, and the time is recorded in the parse checkpoint (which technically includes a tiny bit of execution of the second script tag, but this is neglected and billed to the parser). The `eval` checkpoint is recorded only after your business logic has all been evaluated.

Chrome on Android now offers a much more powerful technique to debug startup time: using `window.performance` to read performance-related timestamps in the code. Figure 5 illustrates how to compute the load time of the page. Whether you are using JavaScript Date objects or the new `window.performance` timestamps, it is a good idea to record this number by making a request to your server side so that you can track page-load metrics as a basic performance indicator of your site—it is so easy to regress that without continuous monitoring, performance is almost sure to degrade over time.

It is highly instructive to look at these numbers to appreciate how bad parse time is, and then to apply tricks to avoid parse latency. The most powerful of these tricks is allowing the browser to load your JavaScript without recognizing it as script and defer parsing and initial evaluation to a time of your choosing.

There are two methods for accomplishing this. The older one is commenting out your JavaScript and using another block of script to uncomment the content of the tag only when it is needed, as described in the Google mobile blog post from 2009.[3] More recently, a cleaner technique has been used for the same purpose. It involves setting the type of the script to an unrecognized type and changing it to text/JavaScript later. In this approach, you start with script blocks that look like this:

```
<script type="deferred" id="module1">
var start _ checkpoint = new Date();
</script>
```

When you are ready to use the script, the JavaScript finds the script tag and modifies the type, which will cause the browser to parse and evaluate it at that time.

By putting your deferred script at the bottom and uncommenting it only when needed, you can very effectively amortize the cost of using JavaScript across the runtime of your application. If your modules are small enough, this technique is cheap enough to be applied at the moment the user first clicks a button or link that needs the script, which can be uncommented, parsed, evaluated, and executed with no noticeable latency.

Avoid Layout and Style Calculation. One of the easiest ways of introducing unexpected latency into a site is inadvertently causing the browser to lay out the document. Seemingly innocuous JavaScript can trigger style recomputation in the midst of execution, while a rearrangement of the same code can be much more efficient.

The rule of thumb is to avoid reading properties that rely on the position of elements on the page to be returned, because any such property could cause the browser to recalculate styles on demand and return the value to your script.

The easiest way to inspect examples of this problem is to use the developer tools in Chrome. Ryan Fioravanti's excellent Google I/O presentation[2] details how to do this by using adb (Android Debug Bridge) to forward a port on a desktop machine to the development-tool port on the Android Chrome browser. Once that is set up, the Events timeline can be used to spot Layout events followed by large gaps of time that represent style recalculation. In the example in Fioravanti's talk, paying attention to when style recalculation was happening made the event handler in question twice as fast as previously.

Monitor Request Size as Well. The focus so far has mainly been on the size of responses to the client and how long the client takes to process data on the client side. Request size can also be a problem, particularly for sites using a large number of cookies alongside every request. This is easiest to monitor on the server side, and ideally all requests made back to the server should be much smaller than a single TCP packet.

Preloading Components is most effective when the user's workflow is almost certain to cause a predictable action just after the initial page loads. For example, in a news site certain articles will have a large click-through rate caused by a combination of placement, content, and imagery, so it makes sense to preload the article page as soon as there is no other work to be done. This can give users a positively delightful experience where the site seems to load instantly.

Another example of this is the AJAX version of the mobile Google Calendar website, where the next day's events are loaded as the user clicks through each day. The effect is the user can easily walk through the week and see the data load instantly. You can compare this to the non-preloaded case by using the month view to jump ahead a few days at a time, exposing how slow it can be to take a round trip to the server to serve the same quantity of data. The round trip involves a noticeable spinner even over Wi-Fi, while the precached data appears instantly, even on old devices.

Optimize Images. Particularly for dynamic images, applying the best compression available will pay off handsomely in overall page speed. You can test image quality on a device to determine if there is a noticeable difference between 50% JPEG quality and 80% or 90%, and note the significant reductions in size that can be achieved. Even dramatic reductions in JPEG quality are often not visible, and the savings can be as much as 40% of the file size. Similar advice applies to .PNG and .GIF files, though these are used less for dynamic content (and therefore they are served from app cache and are not as sensitive to size).

Avoid Inefficient CSS Selectors. Almost every recommendation in Google's "Optimize Browser Rendering" section applies perfectly to mobile, particularly the concerns about needlessly inefficient CSS. There is a trade-off to consider, however: using a complex CSS selector can avoid a Document Object Model (DOM) modification. While complex CSS selectors are considered inefficient by Google, they are fast compared with doing the equivalent work in JavaScript on a mobile device. For other cases where the CSS selector is not being used to avoid JavaScript execution, Google's recommendations apply.

Reduce DNS Lookups. Ideally, a site would load with 0 DNS lookups, as each represents a round trip with the potential to block all other activity. Minimally, however, a DNS lookup will fetch the manifest file, which is done in parallel with page load as the browser brings in resources from the application cache. So it is fairly safe to use the name of your host site and assume it will be in the system's DNS cache whenever the device is online.

If you rely on using multiple hosts to parallelize content loading, you may need one more hostname lookup. There is no point using more than two hosts to parallelize content loading since in the best of cases you are likely to be limited to four simultaneous connections. A second DNS lookup is probably preferable to hardcoding the IP address in your application. This would cause all content to download every time your servers moved and would make it impossible to make good use of any CDN (content delivery network) you may be using to serve data. On balance, the recommendation here is to stick with a single hostname for serving all resources, as the benefit from multiple hostnames to get more parallel downloads is small relative to the implementation complexity, and actual speed gains are unlikely to be significant over poor networks.

By putting your deferred script at the bottom and uncommenting it only when needed, you can very effectively amortize the cost of using JavaScript across the runtime of your application.

Minify JavaScript and CSS. Minification is beneficial for mobile, though the biggest effect is seen in comparing uncompressed sizes. If minification poses a challenge, be sure to compare the sizes only after applying gzip to get a realistic sense of the gains. They may be below 5% and not worth the extra serving and debugging complexity.

Which Recommendations Should Be Used Only Occasionally?

Some recommendations depend on the context of the particular project and should be used only on certain sites.

Use Cookie-Free Domains for Subcomponents. Using cookie-free domains applies to mobile, except that all static content should already be served by the app cache, so the effect is not as strong. If you are not using app cache, then follow this recommendation.

Avoid Empty Image SRC. If you need to create image tags that contain a dummy image, then you can use a data URL to encode a small image in place of an empty string until you can replace the image source.

Keep Components Under 25KB. The 25KB restriction does not apply to resources in the app cache but does apply to everything else. If older devices are important to your site, you may still want to respect this recommendation. If you are designing primarily for future devices, however, only HTML pages still need to be lightweight: the rest of your resources will stay cached in memory until the user restarts the iOS device or forcibly exits the process; and on Android, your components will stay in persistent cache until they expire.

Which Recommendations Should Be Ignored?

Some of the guidelines for desktop development do not apply when building mobile sites.

Do Not Make JavaScript and CSS External. This is one recommendation that cannot really be followed on mobile. Because of a healthy distrust of the mobile device's browser cache policy, it will almost always turn out best if you cache your JavaScript and CSS manually; thus, inlining it all into the main page will deliver the best results in terms of speed. For maintenance reasons, or perhaps if a site has seldom-visited areas where it would be better not to download until first use, you can use XHRs to fetch the Java Script or CSS and the HTML5 database to store the resource for later reuse.

Redirects are a source of round trips and are to be avoided at all costs. Regularly monitoring the response codes from your servers should make it possible to catch redirect mistakes, which are commonly caused by poor URL choices, authentication choices, ads, or external components. It is better to update the site's script and resources and let app cache download and cache all the new locations than to deliver even one redirect return code per user visit. Redirects can also cause unexpected effects with app cache, causing the browser to download the same resources multiple times and store them in the database. This is easiest to test for on the desktop where the sqlite3 command-line tool can be used to look directly at the app-cache database and see what is stored there. On Chrome, an even easier method is to use chrome://appcache-internals to inspect the state of app cache.

Redirects are a source of round trips and are to be avoided at all costs. Regularly monitoring the response codes from your servers should make it possible to catch redirect mistakes.

YSlow also recommends avoiding duplicate script. Certainly after all the effort made to minify, obfuscate, and manually cache your script tags, it will be worth ensuring the browser is not parsing and evaluating the same piece of script twice. Removing duplicate scripts remains a solid piece of advice for mobile sites.

Do Not Configure ETags. In the case of entity tags (ETags), the removal advice is the bit that applies. As you are already storing all the resources in app cache, and app cache is separate per host domain, there is no benefit to using ETags—and the mobile browser cache cannot be trusted to retain the components, anyway.

Do Not Make AJAX Cacheable. Rather than trusting the browser to cache AJAX responses, it is best to build either a full-blown write-through cache layer or an XHR caching layer on top of the HTML5 local storage facilities.

Do Not Split Components Across Domains. I have already discussed parallel downloads, and the opportunity for leveraging high-bandwidth connections on mobile is too limited to make this a core technique.

Reducing cookie sizes applies to mobile just as it does to the desktop. Specifically, be sure the cookies for a page do not, in aggregate, cause any requests to be split across multiple packets. Aim for 1,000 bytes or less of cookie data.

Do Not Choose `<link>` over `@Import`. Despite the existing recommendation stated in the title here, for absolute speed on mobile, neither `link` nor `@import` is appropriate. Instead, inline styles in the main body of the page create the fastest load time and the most app-like Web experience. If separate resources are a must to simplify serving, the preference should be `<link>` to avoid rendering content with missing style information.

Do Not Pack Components into a Multipart Document. As mobile devices do not support multipart documents, app cache is required.

What Still Applies?

With these extensive modifications to the guidelines, the YSlow and Page Speed recommendations may seem to have very little application to mobile Web development. Quite a few recommendations can still be used effectively for mobile, however, depending on your exact requirements. Here is a list of recommendations from YSlow that can still be useful for mobile sites:

- Use a CDN
- Add an expires or cache-control header
- Put style sheets at the top
- Avoid redirects
- Remove duplicate scripts
- Flush the buffer early
- Use GET for AJAX requests
- Postload components

- ▸ Reduce the number of DOM elements
- ▸ Minimize the number of iframes
- ▸ No 404s
- ▸ Reduce cookie size
- ▸ Optimize CSS sprites
- ▸ Do not scale images in HTML
- ▸ Make favicon.ico small and cacheable

Similarly, Google's PageSpeed recommendations, except those that have been specifically modified in this article, can apply to mobile. These tools from Yahoo! and Google should be the first line of defense against a slow site.

Putting It All Together

Let's spend a moment on how to measure performance. One technique already referenced is using adb to connect to Chrome running on your mobile device. This enables you to apply the full set of Web development tools in Chrome to the instance running on your phone. For interactive debugging, this cannot be beat. It is worth spending some time to understand the Network, Timeline, Profile, and Resources screen so you can master debugging and optimizing using Chrome. Since Chrome and Mobile Safari are very similar rendering engines, this will pay dividends on both major mobile platforms.

For the long run, do not forget to implement server-side statistics. Using JavaScript timestamps or `window.performance`, you can measure and track all the key load-time latency metrics so you know when there is a regression and in which part of the system it is. Frequently authentication, domain name changes, or third-party components wind up introducing additional redirects or network traffic that are not noticeable on good connections but that show up for your end users' real-world networks.

For AJAX, it is generally valuable to build a generic AJAX fetch layer that knows how to cache AJAX requests persistently. For example, a simple technique for a news or discussion board site would be to use unique hashes for each piece of content and a generic AJAX fetch layer that records all incoming data in an LRU (least recently used) cache in local storage. This general pattern of unique URLs and a simple-minded cache scheme will make dynamic content caching easy to understand and easy to implement and reuse across multiple sites.

For the final bit of polish, it is worth thinking hard about using CSS tricks to prerender layers of your site and transition between them. By rendering dialogs, for example, at a low opacity and fading or sliding them in as needed, you can create application-like experiences that run smoothly in response to user actions.

For the ultimate fit and finish, the site should never let the browser paint elements in an order of its choosing, but instead use layers of divs that are revealed in the right order and only when completely rendered. For transition durations, values of 150ms–300ms strike a nice balance between a snappy transition and a slick looking one—slower transitions look better but cost too much time, and faster ones look choppy.

A final note to consider is whether to use any application meta tags to give your website a special icon on the desktop, a splash screen, or to load full screen. For some sites these little touches add significantly to the fit and finish. The main drawback of these methods is that on iOS the Web view used for full-screen mode does not take advantage of the JavaScript JIT (just-in-time) compiler in Webkit, so JavaScript compute-intensive code runs slower. This will almost never affect the site, but you will need to test for it. In addition, the cookies are separate from the browser cookies, which could mean authenticating multiple times for different bookmarks.

On iOS, Mobile Safari gets special treatment and can start up significantly faster than a full-screen bookmark. Given these considerations, if you want to use bookmarking, there is sample code that provides a nice example of how to show users a different screen on first load, encouraging them to bookmark your site to their home screens.

Conclusion

With diligent attention to existing desktop recommendations and a few additions that take into account mobile network and CPU speed challenges, it is possible to create very fast, very slick website experiences for users. If you cannot follow all the recommendations, this article has presented the most valuable techniques first—namely, HTML5 app cache and deferred JavaScript techniques. If you can follow every recommendation, then your users can look forward to a fast mobile experience that they will come back to again and again.

Acknowledgments

Almost every idea in this article is not mine, but rather the ideas of many Google teammates working on mobile websites over the years. They are Matthew Bolohan, Bikin Chiu, Ryan Fioravanti, Andrew Grieve, Alex Kennberg, Pavel Kobyakov, Robert Kroeger, Peter Mayo, Joanne McKinley, Derek Phillips, Keith Stanger, Neil Thomas, and Ilia Tulchinsky.

Related articles on queue.acm.org

Streams and Standards: Delivering Mobile Video
Tom Gerstel
http://queue.acm.org/detail.cfm?id=1066067

Mobile Media: Making It a Reality
Fred Kitson
http://queue.acm.org/detail.cfm?id=1066066

Mobile Devices in the Enterprise: CTO Roundtable Overview
Mache Creeger
http://queue.acm.org/detail.cfm?id=2019556

References

1. Cisco. Cisco Visual Networking Index: global mobile data traffic forecast update, 2012–2017 (2013); http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.
2. Fioravanti, R. Building high-performance mobile Web applications; http://rjf-io2012.appspot.com/#5; video: http://www.youtube.com/watch?v=jD_-r6y558o.
3. Google Code Blog. Gmail for Mobile HTML5 Series: reducing startup latency; http://googlecode.blogspot.ca/2009/09/gmail-for-mobile-html5-series-reducing.html.
4. Google Developers. Minimize round-trip times (2012); https://developers.google.com/speed/docs/best-practices/rtt.
5. Google Developers. Web performance best practices (2012); https://developers.google.com/speed/docs/best-practices/rules_intro.
6. Grove, R. Mobile browser cache limits, revisited; http://www.yuiblog.com/blog/2010/07/12/mobile-browser-cache-limits-revisited/.
7. Souders, S. Mobile cache file sizes; http://www.stevesouders.com/blog/2010/07/12/mobile-cache-file-sizes/.
8. Yahoo! Developer Network. Best practices for speeding up your Web site (2013); http://developer.yahoo.com/performance/rules.html.

Alex Nicolaou is Chrome Engineering Manager at the Google office in Waterloo, Ontario where he works on building ChromeOS for ARM platforms. Prior to joining Google in 2006, he was president of aruna.ca Inc., a startup developing RDBMS based on text-search algorithms and data structures and was part of LiquiMedia Inc.

DOI:10.1145/2492007.2492024

Article development led by acmqueue
queue.acm.org

An overview of techniques to speed page loading.

BY TAMMY EVERTS

Rules for Mobile Performance Optimization

PERFORMANCE HAS ALWAYS BEEN CRUCIAL to the success of websites. A growing body of research has proven that even small improvements in page-load times lead to more sales, more ad revenue, more stickiness, and more customer satisfaction for enterprises ranging from small e-commerce shops to megachains such as Walmart.

For years Web developers could count on steady improvements in hardware and bandwidth to help deliver an optimal user experience. In recent years, however, the explosion of mobile Web browsing has reversed this. The lower bandwidth, higher latency, smaller memories, and lower processing power of mobile devices have imposed an even more urgent need to optimize performance at the front end in order to meet user expectations.

This article summarizes the case for front-end optimization and provides an overview of strategies and tactics to speed up your pages, with an emphasis on addressing mobile performance issues.

No matter how interesting, beautiful, or cleverly interactive your Web pages are, if they take more than two or three seconds to render, whether on a desktop or a mobile device, users quickly become impatient. They are measurably less likely to convert from browsing to buying and may even hit the back button or close the browser before the page ever loads.

Even delays of less than one second significantly affect revenues. In 2006 Marissa Mayer, with Google at the time, recounted that, after users indicated they wanted to see more than 10 search results per page, Google experimented with showing 30 instead. To Google's surprise, traffic and revenue dropped by 20% in this experiment, apparently

IMAGE FROM SHUTTERSTOCK.COM

because the pages with more results took just an extra half-second to load.[5]

User expectations have only escalated since then. A 2009 study by Forrester Research on behalf of Akamai identified two seconds as the threshold for acceptable Web-page response times and found that 40% of consumers abandon a page that takes longer than three seconds to load. Just one year later, another study done for Akamai found that the number of users who abandon a page after three seconds had risen to 57%.[1,7]

Furthermore, users on mobile devices expect performance to be at least as good as—if not better than—what they experience on their desktop. The Harris Interactive 2011 Mobile Transactions Survey, commissioned by Tealeaf Technology (now IBM), reported that 85% of adults who had conducted a mobile transaction in the previous year expected the mobile experience to be equal to or better than shopping online using a laptop or desktop computer, and 63% said they would be less likely to buy from the same company via other channels if they experienced a problem conducting a transaction on their mobile phones.[10] In other words, poor mobile performance hurts companies on all other platforms, including bricks-and-mortar.

Mobile traffic is expanding rapidly. For many consumers, their phone or tablet has become their primary portal to the Internet, but performance is falling short of expectations. A study published by Equation Research on behalf of Compuware in February 2011 found that almost half (46%) of mobile users said websites load more slowly than expected on their phones. Nearly 60% expect pages to load in three seconds or less, and 74% report they would leave a site if a single page took five seconds or more to load. A 2012 study of 200 leading e-commerce sites by Strangeloop Networks (now Radware) found that the median load time was 11.8 seconds over 3G (see Figure 1); performance over LTE fared only slightly better, at 8.5 seconds.[8]

Three Limiting Factors for Mobile Performance. As already mentioned, mobile devices have inherent performance limitations: lower bandwidth, smaller memories, and lower processing power. These challenges are compounded by external issues, notably:

Web pages are bigger than ever. According to the HTTP Archive, the average Web page carries a payload of more than 1MB and contains at least 80 resources such as images, Java Script, CSS (Cascading Style Sheets) files, etc. This has a significant impact on desktop performance. Its impact on mobile performance—and particularly on 3G performance—is much more dramatic. This impact will be felt even more keenly over the next three years. At the current rate of growth, pages could surpass 2MB by 2015.

Latency can vary widely. It can range from as little as 34ms for LTE to 350 ms or more for 3G. Mobile latency is consistent only in its inconsistency, even when measured at the same location. This is due to a number of variables beyond the amount of data passing through the tower. Factors such as the weather, and even the direction the user is facing, can have a significant impact.

Download speeds can also experience huge variance. The speeds can range from a mere 1Mbps over 3G to as much as 31Mbps over LTE. It is interesting to compare this to the average U.S. broadband speed of 15Mbps, and to note that 3G can be up to 15 times slower than broadband, while LTE can be up to twice as fast.

M.Sites are not a Cure-All for Mobile Performance Pains. Many site owners attempt to respond to the combination of high user demands, large Web pages, and poor connection speeds by developing smaller, faster, stripped-down m.sites; however, these attempts are not completely effective, as up to 35% of mobile users will choose to view the full site when given the option.

These full-site visitors are significantly more likely to spend than m.site visitors. One study found that for every $7.00 of mobile-generated revenue, $5.50 was generated via full site. Only $1.00 came via m.site, and $0.50 via app.[9]

Addressing the Problem. The chief strategies for improving site performance have not changed as usage has migrated from the desktop to mobile phones and tablets, although a few new tactics have emerged.

Only 20% of the time required to display a typical Web page, whether in a desktop or mobile browser, is consumed by loading the page's HTML. The remaining 80% is spent loading the additional resources needed to render the page—including style sheets, script files, and images—and performing client-side processing.

The three main strategies for improving performance are:

- Reducing the number of HTTP requests required to fetch the resources for each page.
- Reducing the size of the payload needed to fulfill each request.
- Optimizing client-side processing priorities and script execution efficiency.

Because mobile networks are usually slower than those available to desktop machines, reducing requests and payloads takes on huge importance. Mobile browsers are slower to parse HTML and execute JavaScript, so optimizing client-side processing is crucial. In addition, mobile browser caches are much smaller than those of desktop browsers, requiring new approaches to leveraging local storage of reusable resources.

The remainder of this article summarizes tactics you can use to address these challenges. While automated tools are available for most of these practices, many can also be implemented manually (by an experienced front-end developer). It is crucial to note that an overarching challenge with the manual implementation of many of these techniques is control of resources. Often in CMS (content management system) or other Web applications, pages can include HTML snippets, CSS, and JavaScript files that are either generated or hosted off-site, meaning developers do not have access to optimize them.

Reduce Requests

The biggest drain on performance is usually the need to complete dozens of network round-trips to retrieve resources such as style sheets, scripts, and images. This is especially true with the relatively low bandwidth and high latency of mobile connections. CDNs (content delivery networks) can help a bit by bringing content geographically closer to users, but the number of requests has a much greater impact on page-load times than the distances those requests travel. In addition, recent findings suggest CDNs have limited effectiveness for mobile users.[3]

Here, I discuss several approaches to minimizing HTTP requests.

Consolidate Resources. By now it is standard practice for developers to consolidate JavaScript code and CSS styles into common files that can be shared across multiple pages. This technique simplifies code maintenance and improves the efficiency of client-side caching.

In JavaScript files, be sure that the same script is not downloaded multiple times for one page. Redundant script downloads are especially likely when large teams or multiple teams collaborate on page development. It might surprise you how often this occurs.

Spriting is a CSS technique for consolidating images. Sprites are simply multiple images combined into a rectilinear grid in one large image. The page fetches the large image all at once as a single CSS background image and then uses CSS background positioning to display the individual component images as needed on the page. This reduces multiple requests to only one, significantly improving performance.

Ease of implementation: Moderate, but requires having access to resources. Depending on the level of control the developer has over the website, some resources may not be able to be consolidated (for example, if they are generated by a CMS). Also, some resources may be located on external domains, which can cause problems for consolidation. It is also important to note that resource consolidation can be a double-edged sword for mobile browsers. Reducing requests improves performance the first time, but larger consolidated resources may not be cached efficiently, so be careful to combine consolidation techniques with other techniques to optimize `localStorage`.

Use Browser Caching and `LocalStorage`. All modern browsers use local memory to cache resources that have been tagged with Cache-Control or Expires headers that indicate how long the item can be cached. In addition, ETag (entity tag) and Last-Modified headers indicate how resources should be repopulated in the cache after they have expired. The browser fetches cached items locally whenever possible, avoiding unnecessary server requests, and it flushes items that have expired or have not been used recently when cache space runs short. The resources stored in browser object caches commonly include images, CSS, and JavaScript code, and caching is essential to achieving acceptable site performance. (A separate cache holds entire rendered pages to support use of the Back and Forward buttons.)

Mobile browser caches, however, are usually much smaller than those on desktop machines, causing items to be flushed quickly. The HTML5 Web storage specification provides a great alternative to relying only on browser caching. The HTML5 `localStorage` JavaScript object has been implemented in all the major desktop and mobile browsers. Script code can easily check for support of HTML5 `localStorage` and then use it, if available, to save and read key/value data, usually 5MB per domain. This capability makes `localStorage` very well suited for client-side caching, although read/write speeds do vary for different mobile browsers. It is usually significantly faster to retrieve a resource from `localStorage` than to request it from a server, and it is more flexible and reliable than relying only on cache headers and the limited browser cache storage available on most mobile devices. In addition, this is one area where mobile browsers are currently ahead of the desktop in efficiency—`localStorage` performance has lagged in desktop implementations where using the standard browser cache may still be the best option.

Ease of implementation: Advanced. While the `localStorage` mechanism may be simple to use, building a cache around it does create some complexities. You will need to take into account all of the issues that a cache handles for you, such as cache expiry (when do you remove items?), cache misses (what if you expected something to be in `localStorage` and it is not?), and what to do when the cache is full.

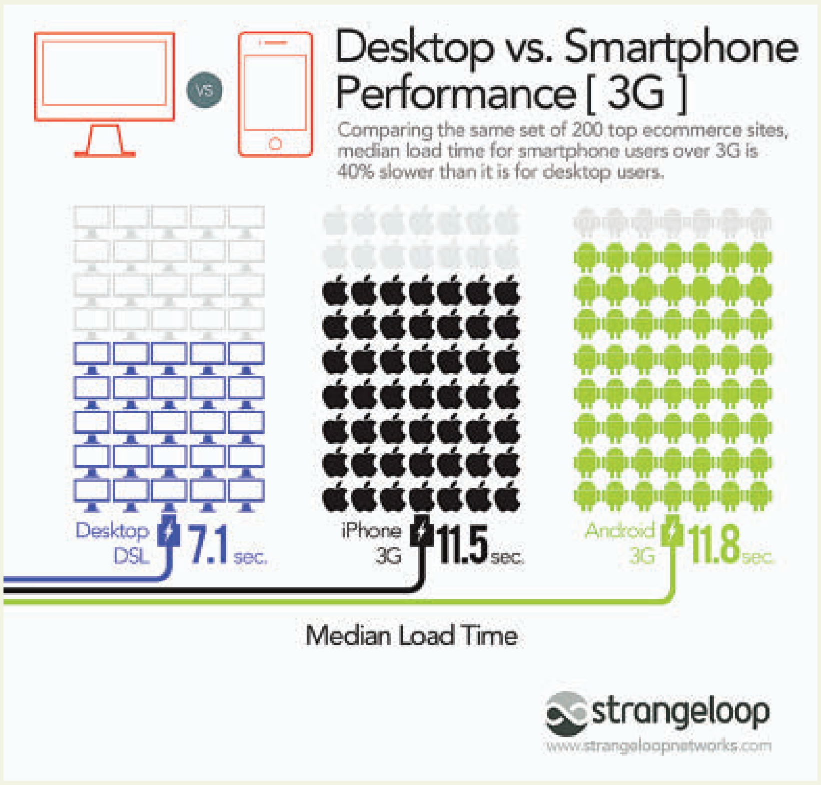
Embed Resources in HTML for First-time Use. The standard pattern in HTML is to include links to external resources. This makes it easier to maintain these resources as files on the server (or in a CDN) and to update them at the source rather than in each of many pages. This pattern also supports browser caching by allowing cached resources to be fetched automatically from the cache rather than from the server, as previously discussed.

For resources that are not already cached in the browser or in `localStorage`, however, this pattern of linking to external resources has a negative impact on performance. A typical page can require dozens of separate requests in order to gather the resources needed to render the page. So, from a performance standpoint, if a resource does not have a high likelihood of already being cached, it is often best to embed that resource in the page's HTML (called *inlining*) rather than storing it externally and linking to it. Script and style tags are supported in HTML for inlining those resources, but images and other binary resources can also be inlined by using data URIs that contain base64-encoded text versions of the resources.

The disadvantage of inlining is that page size can become very large, so it is crucial for a Web application that uses this strategy to be able to track when the resource is needed and when it is already cached on the client. In addition, the application must generate code to store the resource on the client after sending it inline the first time. For this reason, using HTML5 `localStorage` on mobile devices is a great companion to inlining.

Ease of implementation: Moderate. This technique requires the site to have a mechanism to generate a different version of the page based on whether or not the user has visited that page before.

Figure 1. Median load times for desktop and mobile devices.



Use HTML5 `Server-Sent` Events. Web applications have used various polling techniques to update pages continually by requesting new data from a server. The HTML5 EventSource object and Server-Sent events enable JavaScript code in the browser to open a much more efficient unidirectional channel from the server to the browser. The server can then use this channel to send data as it becomes available, eliminating the HTTP overhead of creating multiple polling requests. This is also more efficient than HTML5 WebSockets, which is a richer protocol for creating a two-way channel over a full-duplex connection when a lot of client-server interaction is called for, such as in messaging or gaming.

Ease of implementation: Advanced. This technique is very implementation specific. If your site is currently using other AJAX or Comet techniques for polling, then converting to use Server-Sent events may take quite a bit of recoding the site's JavaScript.

Eliminate Redirects. When users attempt to navigate to a standard desktop site from a mobile device, the Web application often will read the user-agent HTTP header to detect the request is from a mobile device. The application then can send an HTTP 301 (or 302) response with an empty body and a Location header, redirecting the user to the mobile version of the site as required. However, the extra round-trip to the client and back to the mobile site often consumes several hundred milliseconds over mobile networks. Instead, it is faster to deliver the mobile Web page directly in response to the original request, rather than delivering a redirect message that then requests the mobile page.

As a courtesy to users who prefer to view the desktop site even on their mobile devices, you can provide a link on the mobile site that signals your application to suppress this behavior.

Ease of implementation. While this technique is easy in theory, it may not always be possible to put into practice. Many sites redirect to a different server for their m.sites, since those may be hosted elsewhere. Other sites send cookies with the redirect to tell the Web application that they are mobile once they redirect. This may be more difficult to control, depending on the Web application.

Reduce Payload

Size matters. Smaller pages render faster, and smaller resources are fetched faster. Reducing the size of each server response does not usually help performance as much as reducing the number of responses needed for each page. Several techniques, however, do yield a net benefit for performance, especially on mobile devices where bandwidth and processing power must be managed wisely.

Compress Text and Images. Compression technologies such as gzip reduce payloads at the slight cost of adding processing steps to compress on the server and decompress in the browser. These operations are highly optimized, however, and tests show that the overall effect is a net improvement in performance. Text-based responses, including HTML, XML, JavaScript Object Notation (JSON), JavaScript, and CSS, can all be reduced in size by as much as 70%.

Browsers announce their decompression capabilities in the Accept-Encoding request header, and they perform decompression automatically when servers signal that a response is compressed in the Content-Encoding response header.

Ease of implementation: Easy. All modern Web servers will support compressing responses if correctly set up. However, there are still desktop security tools that will remove the Accept-Encoding headers from requests, which will prevent users from getting compressed responses even though their browsers support it.

Minify code. *Minification*, which is usually applied only to scripts and style sheets, eliminates inessential characters such as spaces, newline characters, and comments. Names that are not publicly exposed, such as variable names, can be shortened to just one or two characters. A correctly minified resource is used on the client without any special processing, and file-size reductions average about 20% and style blocks within HTML pages can also be minified. There are many good libraries available to perform minification, often along with services to combine multiple files into one, which additionally reduces requests.

Minification not only reduces bandwidth consumption and latency, but also may mean the difference between a cacheable object and one that is too big for the cache on a particular mobile device. Gzip compression is no help in this regard, because objects are cached by the browser after they have been decompressed.

Ease of implementation: Easy. The Closure Compiler from Google does an incredible job of understanding and minifying JavaScript. CSS minification is a little more troublesome as there are so many CSS hacks for different browsers that can easily confuse minifiers or no longer work correctly after minification. Also note there have been published reports of minification breaking pages, even though the removed characters should not be essential. So be sure to perform functional tests on any pages where you apply this technique.

Resize Images. Images often consume the majority of the network resources required to load Web pages, and the majority of the space required to cache page resources. Small-screen mobile devices present opportunities for speeding transmission and rendering by resizing images. High-resolution images waste bandwidth, processing time, and cache space if the user will be viewing the images only in a small mobile browser window.

To speed up page rendering and reduce bandwidth and memory consumption, dynamically resize images in your application or replace images with smaller versions for mobile sites. Don not waste bandwidth by relying on the browser to scale a high-resolution image into a smaller width and height.

Another option is to load a very low-resolution version of an image initially to get the page up as quickly as possible and then replace that with a higher-resolution version on the `onload` or ready event after the user has begun interacting with the page.

Ease of implementation: Advanced, especially for highly dynamic sites.

Simplify Pages with HTML5 and CSS 3.0. The HTML5 specification includes new structural elements, such as `header`, `nav`, `article`, and `footer`. Using these semantic elements yields a simpler and more efficiently parsed page than using generic nested div and span tags. A simpler page is smaller and loads faster, and a simpler DOM (Document Object Model) means faster JavaScript execution. The new tags are quickly being adopted in new browser versions, including mobile browsers, and HTML5 was designed to degrade gracefully in browsers that do not yet support it.

HTML5 input elements in forms support lots of new attributes that enable declarative HTML code to implement features that previously required JavaScript. For example, the new placeholder attribute can specify instructional text that appears until a user makes an entry, and the new autofocus attribute can specify which input should automatically get the initial focus.

There are also several new types of input elements, which automatically implement commonly needed features without JavaScript. The new types include email, URL, number, range, date, and time, which are efficiently rendered as complex controls with friendly user interfaces and validation. In mobile browsers, the pop-up keyboards often automatically provide keystroke choices appropriate to the specified input type when text input is required. Browsers that do not support the specified input type will simply display a text box.

In addition, new CSS 3.0 features can help create lightweight pages by providing built-in support for gradients, rounded borders, shadows, animations, transitions, and other graphical effects that previously required images to be loaded. These new features can speed up page rendering.

A number of websites offer regularly updated lists showing which features are supported by which desktop and mobile browsers (for example, http://caniuse.com/ and http://mobilehtml5.org/).

Ease of implementation: Advanced. Making these changes manually is extremely complex and time consuming, if not impossible. If you use a CMS, it may generate a great deal of HTML and CSS that you have no control over.

The order in which a browser executes the various steps needed to construct a page can have a major impact on performance, as do the complexity of the page and the choice of JavaScript techniques.

Optimize Client-Side Processing

The order in which a browser executes the various steps needed to construct a page can have a major impact on performance, as do the complexity of the page and the choice of JavaScript techniques. This is especially true on mobile devices where client-side processing is constrained by slower CPUs and less memory. The following sections provide some tactics for increasing the efficiency of page processing.

Defer Rendering Below-the-Fold Content. You can ensure the user sees the page quicker by delaying the loading and rendering of any content that is below the initially visible area, sometimes called "below the fold." To eliminate the need to reflow content after the remainder of the page is loaded, replace images initially with placeholder `<img>` tags that specify the correct height and width.

Ease of implementation: Moderate. Some good JavaScript libraries are available that can be used for below-the-fold lazy image loading.[12]

Defer Loading and Executing Scripts. Parsing JavaScript can take up to 100 milliseconds per kilobyte of code on some mobile devices. Many script libraries are not needed until after a page has finished rendering. Downloading and parsing these scripts can safely be deferred until after the `onload` event. For example, scripts that support interactive user behavior, such as drag and drop, cannot possibly be called before the user has even seen the page. The same logic applies to script execution. Defer as much as possible until after `onload` instead of needlessly holding up the initial rendering of the important visible content on the page.

The script to defer could be your own or, often more importantly, script from third parties. Poorly optimized scripts for advertisements, social media widgets, or analytics support can block a page from rendering, sometimes adding precious seconds to load times. Also, carefully evaluate the use of large script frameworks such as jQuery for mobile sites, especially if you are using only a couple of objects in the framework.

Ease of implementation: Moderate. Many third-party frameworks now provide deferred or async versions of their APIs. The developer just has to switch to these new versions. Some JavaScript may be more complex to defer as there are many caveats to running scripts after `onload` (for example, what do you do if you have a script that wants to attach to the `onload` event? If you defer it after `onload`, it has missed its chance).

Use AJAX for Progressive Enhancement. Asynchronous JavaScript and XML (AJAX) is a technique for using the XHR (`XMLHttpRequest`) object to fetch data from a Web server without refreshing the page where the code is running. AJAX enables a page to display updated data in a section of a page without reconstructing the entire page. This is often used to respond to user interaction, but it can also enable your application to load a bare-bones version of a page quickly, and then to fill in more detailed content while the user is already viewing the page.

Despite the name, `XMLHttpRequest` does not tie you to using only XML. You can call its `overrideMimeType` method to specify "application/json" and work with JSON instead of XML. Using `JSON.parse` is up to twice as fast and more secure than using the generic `eval()` function.

Also, remember that AJAX responses will benefit from many of the same optimization techniques recommended for standard responses. Be sure to apply cache headers, minification, gzip compression, resource consolidation, among others, to your AJAX responses.

Ease of implementation: Difficult to quantify, as this technique is very application specific. Because of cross-domain issues, you would need to use XHR2, as well as control the external domain to make cross-domain XHR requests.

Adapt to the Network Connection. Especially with mobile networks that may charge extra for using more bandwidth, certain techniques should be used only when combined with code to detect the type of connection. For example, preloading resources in anticipation of future requests is usually smart, but it may not be a responsible strategy if the user's bandwidth is metered and some of those resources may never be needed.

On Android 2.2+, the *navigator.connection.type* property returns values that allow you to differentiate Wi-Fi from 2G/3G/4G connections. On Blackberry, *blackberry.network* provides similar information. In addition, server-side detection of User-Agent header data or other information embedded in requests can alert your application to the quality of the connection in use.

Ease of implementation: Advanced. The Network Information API has changed recently.[11] Rather than defining the network as Wi-Fi, 3G, and so on, it now gives information about the bandwidth, with examples such as "very-slow, slow, fast and very-fast." There is a property that tries to tell the estimated MB/s, and a Boolean "metered" measurement that does its best to be correct, but this is very difficult for a browser to determine. Measuring somewhere and adapting is probably still the best idea but is quite challenging.

Use the HTML5 Web Worker Spec for Multithreading. The Web Worker specification in HTML5 introduces multithreaded concurrent execution to the world of JavaScript programming. In addition, this particular implementation of multithreading eliminates problems that have plagued developers working with multiple threads on other platforms—specifically, problems that occur when one thread modifies a resource that is also being used by another thread. In Web Worker code, spawned threads cannot access the resources of the main user-interface (UI) thread.

AJAX enables a page to display updated data in a section of a page without reconstructing the entire page.

For improving the performance of mobile sites, Web Worker code can be valuable for preloading resources that a user is likely to need to complete future actions, especially when the user's bandwidth is not metered. With the limited processor capabilities of mobile devices, extensive preloading can interfere with UI responsiveness in the current page. Using multithreaded code that employs Web Worker objects (and possibly `localStorage` to cache the data), operations that preload resources can execute on a separate thread without impacting current UI performance.

Note that the Web Worker spec, while implemented in Android since 2.0, was not supported on the iPhone until iOS 5. On the desktop, Internet Explorer was the laggard, adding support for Web Worker only in IE 10.

Ease of implementation: Moderate. While this technique is not incredibly difficult to implement, there are some restrictions to Web Workers that make them difficult to find places for. They do not have access to the page's DOM and cannot modify anything on the page. Making this practice work requires a very specific type of background calculation or process that fits well as a background Web Worker.

Replace Click Events with Touch Events. On touchscreen devices, the `onclick` event does not fire immediately when a user taps the screen. Instead, the device waits up to half a second (300 milliseconds on most devices), giving the user a chance to initiate some other gesture rather than a click. This delay, however, can significantly impede the responsive performance that users expect. To fix this, use the `touchend` event instead. That event fires immediately when the user taps the screen.

To ensure the user does not experience unexpected behavior, you may also want to use the `touchstart` and `touchmove` events. For example, do not assume that `touchend` on a button means click unless there was also a `touchstart` event on the button—not if the user touched somewhere else and dragged to the button before ending the touch. You could use a `touchmove` event after `touchstart` to prevent treating the following `touchend` as a click, assuming the moving gesture was not intended to be a click.

In addition, you may still want to handle the `onclick` event to ensure the browser changes the appearance of the button to show a clicked state, and to support browsers that do not handle touch events. To avoid duplicate code execution when both `touchend` and `onclick` code fire, add a click event handler that calls `preventDefault` and `stopPropagation` if the click was the result of a user tap that was already handled by `touchend`.[4]

Ease of implementation: Advanced. This technique requires much more work to add and maintain links on a page. The code testing for touch events must be resilient against gestures that may be happening instead of a click, such as a zoom or swipe.

Support the SPDY Protocol. Some of the performance bottlenecks that afflict websites, whether desktop or mobile, result from inefficiencies in the application-layer HTTP and HTTPS protocols. In 2009, Google began work on an alternative protocol named SPDY (pronounced "speedy") that addresses some of these limitations. The goal is to make this an open source project that will be implemented by multiple browsers and Web servers, but initially it was supported only in Google's Chrome browser (in version 10 or later) and on Google sites. As Web servers are released that implement SPDY, sites will be able to use this protocol for any user with a browser that supports it. In a test implementing SPDY on a representative group of 25 of the top 100 Internet sites, Google observed speed improvements from 27% to 60%.[2]

SPDY automatically uses gzip compression on all content, and unlike HTTP, it also uses gzip on header data. SPDY employs multiplexing technology to enable multiple streams of requests of responses to be sent over a single TCP connection. In addition, SPDY allows requests to be prioritized, so, for example, a video that is central to a page's content can be given a higher priority than an advertisement in the margin.

Perhaps the most revolutionary innovation in SPDY is that streams can be bidirectional and can be initiated by either the client or the server, allowing content to be pushed to clients without first being requested. For example, when a user first visits a site, and therefore does not yet have any of the site content cached, the server can push all required resources in response to the first page request instead of waiting for each resource to be separately requested. As an alternative, the server can send hints to the client, suggesting resources that will be needed, but still allowing the client to initiate the requests. This is still faster than waiting for the client to parse the site pages and discover the resource requirements on its own.

Although SPDY is not specific to mobile platforms, the limited bandwidth available over mobile networks means SPDY optimizations will be especially useful in reducing latency for mobile sites when supported.

Ease of implementation: Moderate to advanced, depending on the site and server environment. Google has a SPDY module for Apache 2.2—mod_spdy—that is available for free; however, mod_spdy has threading model issues and does not play well with mod_php by default, so this requires additional attention in order to ensure it is running correctly on your site.[6]

Do Not Forget to Test!

No discussion of performance optimization would be complete without a reminder that continuous and careful testing is essential. Every change to your system is just a theory until it is tested against a baseline. Guessing where performance bottlenecks occur is meaningless unless based on real test data.

Great open source and commercial tools are available to provide synthetic tests, complete with geographical distribution and bandwidth/latency throttling. In addition, real-user monitoring (RUM) tools take testing out of the lab and into the field of unpredictable user behavior.

Look for testing options that support mobile, as well as desktop scenarios. If you choose an automated solution, be sure to choose one that continually tests and refines the optimizations it applies.

Performance optimization cannot be effective if it is merely a single step in a linear development process. Rather, it must become part of an ongoing cycle of continuous improvement. C

Related articles on queue.acm.org

Mobile Application Development: Web vs. Native
Andre Charland and Brian LeRoux
http://queue.acm.org/detail.cfm?id=1968203

The Evolution of Web Development for Mobile Devices
Nicholas C. Zakas
http://queue.acm.org/detail.cfm?id=2441756

Making the Mobile Web Faster
Kate Matsudaira
http://queue.acm.org/detail.cfm?id=2434256

References

1. Bustos, L. Every second counts; how web-site performance impacts shopper behavior. *GetElastic* (2009); http://www.getelastic.com/performance/.
2. Chromium Projects. SPDY: an experimental protocol for a faster Web; https://sites.google.com/a/chromium.org/dev/spdy/spdy-whitepaper.
3. Everts, T. Case study: How effective are CDNs for mobile visitors. *Web Performance Today*; http://www.Webperformancetoday.com/2013/05/09/case-study-cdn-content-delivery-network-mobile-3g/.
4. Fioravanti, R. Creating fast buttons for mobile Web applications. Google Developers (2011); http://code.google.com/mobile/articles/fast_buttons.html.
5. Linden, G. Marissa Mayer at Web 2.0. Geeking with Greg (2006); http://glinden.blogspot.com/2006/11/marissa-mayer-at-Web-20.html.
6. mod-spdy; http://code.google.com/p/mod-spdy/.
7. PhoCusWright. PhoCusWright/Akamai study on travel site performance; http://connect.phocuswright.com/2010/06/phocuswrightakamai-study-on-travel-site-performance/; http://www.akamai.com/dl/whitepapers/Akamai_PCW_Travel_Perf_Whitepaper.pdf.
8. Radware. Case studies from the mobile frontier: the relationship between faster mobile sites and business KPIS (2011); http://www.strangeloopnetworks.com/resources/research/state-of-mobile-ecommerce-performance/.
9. Radware. 2012 state of mobile e-commerce performance; http://www.strangeloopnetworks.com/resources/videos/case-studies-from-the-mobile-frontier-the-relationship-between-faster-mobile-sites-and-business-kpis-video/.
10. Tealeaf. Report on the Mobile Customer Experience. Based on the Harris Interactive 2011 Mobile Transactions Survey; http://www.tealeaf.com/customer-experience-management/resource-center/rgister.php?doc=mobile-cem.
11. W3C. Network Information API (2012); http://www.w3.org/TR/netinfo-api/.
12. YUI. ImageLoader. Yahoo! User Interface Library; http://yuilibrary.com/yui/docs/imageloader/.

Tammy Everts has worked in user experience and Web performance since 1997. She currently works at Radware, evangelizing performance both in-house and out in the world. Previously, she held roles as research director at Strangeloop Networks and director of user experience at Habanero Consulting. She blogs about performance issues, research, and trends at http://webperformancetoday.com.

DOI:10.1145/2492007.2492025

We should be, for the sake of millions of people with pressing legal needs.

BY MARC LAURITSEN

Are We Free to Code the Law?

THE EMERGENCE OF interactive online services for legal self-helpers has triggered suppression efforts by the legal profession, as well as by state government officials in the U.S. While couched in terms of consumer protection, and at least partly motivated by such concerns, these efforts are also seen by some as blatant turf management by a profession anxious to avoid further erosion of its monopoly over legal advice and representation.

Often neglected in these discussions is whether restricting the distribution of software is within the legitimate scope of government action. No one would contend that attempts to suppress books, pamphlets, and speeches on how the legal system works and what forms one needs to interact with it would pass constitutional muster. Is providing software that helps people meet their legal needs an activity the state can prohibit under the U.S. Constitution?

Here, I explore ways software-based legal-assistance systems can be understood for purposes of public policy and constitutional analysis. The focus is on circumstances in the U.S., but many other countries face the same issues.

Assistance and Authorization

Individuals and organizations who need to prepare documents with legal significance turn to a variety of sources, including form books, courts, government agencies, physical form suppliers,[a] packaged software,[b] online form sites,[c] free online document repositories,[d] notaries public, legal-document technicians, conventional

a See, for example, Blumberg (http://www.blumberg.com)
b See, for example, Turbotax (http://turbotax.intuit.com), Will Maker (http://www.nolo.com/products/quicken-willmaker-plus-WQP.html), and WillWriter (http://www.broderbund.com/p-124-willwriter.aspx)
c See, for example, U.S. Legal Forms (http://www.uslegalforms.com), SmartLegalForms (http://www.smartlegalforms.com), and CompleteCase.com (http://completecase.com)
d See, for example, Docracy (http://www.docracy.com/)

» key insights

- **Online document-preparation services and other forms of automated legal assistance raise concerns about the unauthorized practice of law.**
- **Such concerns should be balanced against social policy and economic freedom.**
- **Software programs are more like books than like personal human services when determining whether they deserve protection under provisions like the First Amendment of the U.S. Constitution.**

ILLUSTRATION BY PHIL WRIGGLESWORTH

private law practices and corporate law departments, and virtual law practices.[3]

An increasingly popular, and controversial, category of service providers generates customer-specific documents over the Internet, using interactive software, without purporting to be engaged in the practice of law, including:

▸ Commercial services;[e]

▸ Nonprofit sites;[f]

▸ Governmental and court sites (such as self-help court resources);[g] and

▸ Free services by law firms.[h]

Most of these services use specialized document-assembly software long used by lawyers themselves; for an overview of document assembly and other specialized technologies used by lawyers, see Lauritsen.[6] That technology enables someone to program "what words go where" under various sets of answers, gathered in interactive questionnaires that change as users work through them, with context-specific guidance. Applications can embody rule sets of arbitrary size and complexity and generate highly tailored and precisely styled documents.

In addition to commercial, governmental, and nonprofit initiatives, courses are offered at a growing number of law schools, some under an "Apps for Justice" rubric, in which students build useful software applications as part of their education, results of which can be made available to the public.[i]

The debate. Consider the following imagined example of the arguments one encounters (sometimes within a single head):

Voice A. At least in my state, these new services are blatantly illegal. By telling people what legal documents they need, and preparing them, they are engaged in the practice of law in all but name.

Voice B. Even if the provider makes perfectly clear it is not practicing law and the user explicitly acknowledges it?

A. We don't think it is OK for unlicensed people to perform medical procedures, just so long as they do not claim to be doctors. Or to manufacture devices for self-help surgery.[j]

B. Cutting yourself open and generating a simple will are not exactly analogous.

A. Would you allow people to extract teeth and fill cavities without a license, so long as they do not claim to be dentists? What about self-help pharmacies that dispense drugs after some interaction with a medical expert system?

B. That's different. Online legal help systems just provide information. Words. They do not do anything physically.

A. An improper legal "procedure" can cause a lot of financial and emotional damage, maybe even result in loss of shelter, child custody, citizenship, or liberty.

B. Lots of things people are allowed to do are dangerous. A weekend do-it-yourselfer can cause real damage with a power saw. Should we bar home-improvement television shows and limit power tools to licensed craftsmen?

A. Law is special. We need lawyers, and it's only fair that in exchange for the years of education they are required to have, and the ethical rules they are required to follow, they get exclusive rights to perform certain kinds of services.

B. Come on. We have a huge population unable to afford legal help. Even unemployed lawyers are unwilling to work at rates low enough, and legal aid is grossly underfunded throughout the U.S.

A. That does not mean vulnerable people should be victimized by companies out to make a quick buck or even by well meaning do-gooders. Software rarely does justice to people's legal needs.

B. Why should consumers incur the inconvenience and expense of hiring a lawyer to create documents that someone else is willing to do inexpensively or free? When they are informed of risks, and prepared to accept them? We are talking willing consumers here. This sounds like the nanny state.

A. We regulate many consumer transactions.

B. It seems to me that writing software is like writing a book, an expressive act that should be protected as speech.

A. Do not try to hide behind the First Amendment. These are not "publications" but services, with people behind them.

B. There are people behind books, too.

A. Yeah, but books don't do anything.

B. Well, they do inform people, and they can be written to give very specific advice for very specific circumstances.

A. That doesn't mean software deserves the same protection as written books.

B. Antipathy to these kinds of applications comes mostly from biased and techno-illiterate policymakers. Many lawyers, judges, legislators, and regulators have little understanding of the nature of computer code. And professionals naturally resist demystification of their expertise.

A. Stuff like this could destroy the legal profession. Is that what you want?

B. Hey, some of my best friends are lawyers. Lawyers just need to learn to compete on the merits. If machines

e See http://www.legalzoom.com, http://www.rocketlawyer.com, http://www.smartlegalforms.com, and http://whichdraft.com

f I-CAN! was created by the Legal Aid Society of Orange County, CA; its E-FILE application, a free Web-based tax-assistance program for low-income workers, has returned more than $233 million to U.S. taxpayers (https://secure.icandocs.org/donor2/icanlegal.asp). LawHelp Interactive, a service of Pro Bono Net, has delivered more than one million customized documents for free (https://lawhelpinteractive.org/ and http://collegeoflpm.org/innovaction-awards/award-winners/2010-innovaction-award-winners/); its contributors and operators arguably risk civil and criminal liability in certain U.S. states under certain interpretations of their rules concerning the unauthorized practice of law.

g See, for example, http://www.courts.ca.gov/selfhelp.htm, http://www.nycourts.gov/courthelp, and http://www.nycourts.gov/courthelp

h See, for example, http://www.goodwinfoundersworkbench.com, https://tsc.orrick.com, http://www.startuppercolator.com, and http://www.wsgr.com/wsgr/display.aspx?sectionname=practice/termsheet.htm

i See, for example, http://www.kentlaw.iit.edu/courses/jd-courses/jd-elective-courses/justice-and-technology-practicum and http://www.law.suffolk.edu/academic/jd/course.cfm?CourseID=571. Courses in which students build interactive legal applications have also been offered at Georgetown Law School and New York Law School; see article "Legal Education Goes High-Tech" http://www.law.com/jsp/nlj/PubArticleNLJ.jsp?id=1202556661527 and http://www.virtual-strategy.com/2012/08/02/neota-logic-ceo-fastcase-50

j The 2012 film *Prometheus* included a scene in which the character played by Noomi Rapace disembowels herself of an alien fetus with the aid of a surgical robot.

can perform better than they can, they should consider another career path. Welcome to capitalism.

And so on…

The questions. The questions here fall into two groups: those about the power of government to regulate automated legal assistance and those about the wisdom of doing so. That is, can government prohibit automated legal assistance, and, if it can, to what extent should it?

Do people have a right to write, read, and run software that embodies ideas about how the law works? To what extent are people free to provide automated legal assistance? Is there a right to receive such assistance? To what extent can government enjoin or punish such provision or receipt? Is the distribution of software that helps people with their legal needs an activity that needs to be "authorized?" What is the right regulatory response? Is it good policy to forbid automated legal assistance? Should lawyers be given a monopoly over legal software, as well as over in-person legal services? In general, what are the appropriate boundaries? What principled lines can we draw in this area?

Unauthorized practice of law. Most states have defined law practice, as well as its unauthorized variants, in statutes and case law. Most such definitions extend to the selection and preparation of documents.

Attorneys General, bar authorities, and private plaintiffs in the U.S. have initiated proceedings against providers of automated legal assistance. Several matters are mentioned here to illustrate.

In the *Parsons*[k] case, the Texas Unauthorized Practice of Law Committee sued two manufacturers of software that helped people prepare wills and other documents, and was granted summary judgment by the court. The case was mooted when the Texas legislature crafted the following statutory exception:

"In this chapter, the 'practice of law' does not include the design, creation, publication, distribution, display, or sale, including publication, distribution, display, or sale by means of an Internet website, of written materials, books, forms, computer software, or similar products if the products clearly and conspicuously state that the products are not a substitute for the advice of an attorney."[l]

Is an occasional harm sufficient reason to forgo the power of modern information technology to make a dent in the vast unmet need for legal assistance?

In the *Reynoso*[m] case, the court found a provider of software for bankruptcy preparation was engaged in UPL, laying stress on the point that websites are "put together by people."

Many state bar committees have opined on this subject; for instance, in March 2010 the Pennsylvania Bar Association Unauthorized Practice of Law Committee concluded as follows:

"It is the opinion of the Pennsylvania Bar Association Unauthorized Practice of Law Committee that the offering or providing [in Pennsylvania] of legal document preparation services as described herein (beyond the supply of preprinted forms selected by the consumer, not the legal document preparation service), either online or at a site in Pennsylvania is the unauthorized practice of law and thus prohibited, unless such services are provided by a person who is duly licensed to practice law in Pennsylvania retained directly for the subject of the legal services."[n]

That is, according to authorities in at least some states many of the services in the section on automated legal assistance are violating the law.

Policy

The case for prohibition. Arguments in favor of disallowing automated legal assistance generally involve protection of the public and of the legal profession:

Protecting the public. Some people will undoubtedly be harmed by automated systems. Defective or incomplete legal assistance can cause significant damage, and it is reasonable to assume such damage is more likely

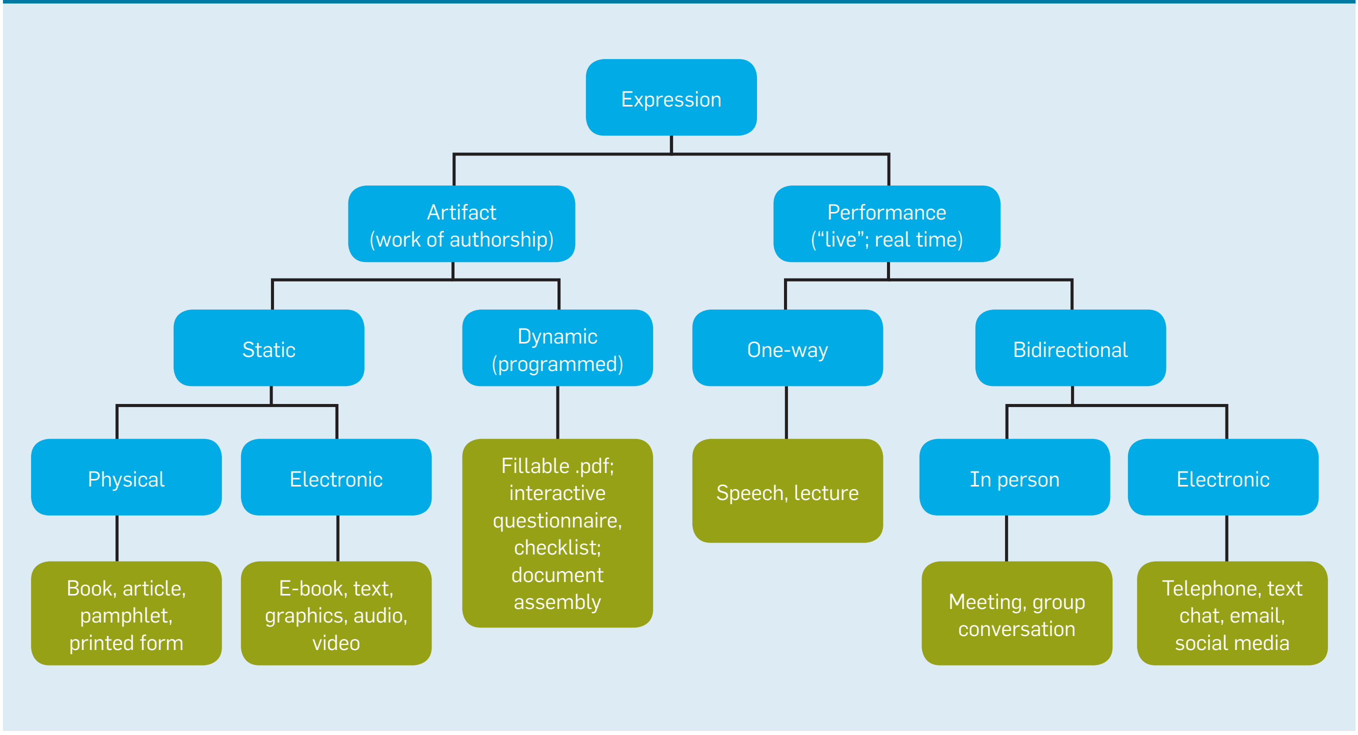
k See *Unauthorized Practice of Law Committee v. Parsons Tech. Inc.*, 1999 Westlaw 47235 (N.D. Tex. Jan. 22, 1999) vacated, 179 F.3d 956 (5th Cir. 1999)

l See Section 81.101(c) of the Texas Government Code

m See in re: *Jayson Reynoso: Frankfort Digital Services et al. v. Sara L. Kistler, United States Trustee et al.* 447 F.3d 1117 (9th Cir. 2007)

n See Pennsylvania Unauthorized Practice of Law Committee, Formal Opinion 2010-01 (Mar. 10, 2010); http://www.pabar.org/public/committees/unautpra/Opinions/2010-01Lgl-DocumentPreparation.pdf

Figure 1. A typology of expressions.


Expression
Artifact (work of authorship)
Performance ("live"; real time)
Static
Dynamic (programmed)
One-way
Bidirectional
Physical
Electronic
Fillable .pdf; interactive questionnaire, checklist; document assembly
Speech, lecture
In person
Electronic
Book, article, pamphlet, printed form
E-book, text, graphics, audio, video
Meeting, group conversation
Telephone, text chat, email, social media

when no lawyer is involved.

Software applications lack common sense. They cannot hear what is not being said. They do not detect nuance or emotion. On the other hand, as with people, they can operate on unspoken assumptions, create the illusion of expertise, and engender unwarranted trust.

Protecting the legal profession. Lawyers are bound by many restrictions on their behavior in exchange for licensing. Is it unfair or unwise to restrict what non-lawyers can do in relation to giving legal advice, counseling, and representation? Part of the societal bargain regarding any profession involves a limited monopoly.

By not allowing unqualified people to advise citizens on their legal affairs, and seeing to it that such advice occurs within appropriately structured and protected relationships, we help ensure the smooth functioning of the legal system and the preservation of an independent legal profession that is so important to democracy.

The case for toleration. Those who favor allowing automated legal-assistance systems generally claim they yield net benefits for both society and the legal profession.

Given the vast amount of textual material already available to legal self-helpers, much of uncertain quality and with few clues as to currency and relevance to specific situations, interactive systems seem more likely to reduce harm than cause it. Their development requires significant time and money few organizations would invest recklessly.

Lawyers themselves are not infallible. Much legal work can be scripted, and software will eventually make fewer mistakes in many contexts. Machines have proven demonstrably better in certain law-related activities (such as coding documents for relevance to pending litigation).[2]

Counterbalanced against the inevitable harms automated assistance sometimes engenders are many clear benefits: more-informed citizens; better-prepared litigants; and cleaner and more-complete documents.

There are also considerations of economic freedom. Business and social entrepreneurs are anxious to innovate in the legal field. Threats of unauthorized practice claims chill innovation. An open market is the best defense against poor quality.

Reaching a balance. Do concerns about harms to consumers and the legal profession outweigh the benefits of citizens having access to legal knowledge through interactive programs? Is an occasional harm sufficient reason to forgo the power of modern information technology to make a dent in the vast unmet need for legal assistance?

The free flow of automated systems seems to offer net advantages. Reasonable regulations should be established to minimize potential harms, but a robust and open market of interactively coded legal ideas is in the best long-term interest of both society and the profession. It is desirable to have lots of such programs competing for use in a free market and to incentivize legal knowledge codification and systemization.

Imagine if a trade union of human "computers"[o] in the 1940s had successfully thwarted the development of electronic machines as the "unauthorized practice of computing." We at least would not, I think, have to worry today about machines doing legal work.

Freedom

Even if a good case could be made for regulating creation and distribution of automated legal-assistance systems, do

o George Dyson's *Turing's Cathedral: The Origins of the Digital Universe*[1] tells the fascinating story of the early days of electronic computing at Princeton's Institute for Advanced Study and elsewhere, including the (non-obstructive) role of human "computers."

such regulations pass muster under the First Amendment?

Admittedly, these applications are novel artifacts not envisioned by the founders.

First Amendment protections are not without exceptions; for instance, they do not authorize people to violate intellectual property or reputational rights. U.S. citizens are not free to engage in libel, copyright infringement, or sedition. Obscenity is only partially protected.

None of these exceptions apply to the expressive activity involved in automated legal-assistance systems.

Alleged misinformation or harmfulness is not viewed as justifying suppression of books, except in extreme circumstances. Government is not appropriately in the business of judging the quality or content of speech. A landmark case[p] held that distributing the 1965 book *How To Avoid Probate* did not constitute the unauthorized practice of law.

One may be inclined to suggest that some automated systems are a form of "commercial speech" and thus deserve less protection. Commercial speech has generally been understood as the activity of beckoning business, not the substantive content of what is being offered. Selling a book does not render it any less deserving of First Amendment protection than giving it away for free.

An alternative way to avoid First Amendment issues is to conclude that programs are not "speech" at all but a form of conduct, analogous to the work of manual document preparers. This involves distinguishing between "pure" speech and "speech plus" that entails actions, as well as words. Sometimes speech-related action is not protected if it is physically dangerous. Does such a dangerousness rationale extend to communicative action?

Several legal scholars have tentatively concluded for the unconstitutionality of repressing online legal services under the guise of the unauthorized practice of law.[q] The following sections lay out an analytical framework that may support more definitive conclusions.

A typology of expressions. Figure 1 outlines one way to organize the varieties of expression a legal self-helper might access; blue boxes are categories, and green boxes contain examples.

Expressive conduct falls into two main categories: creating artifacts, or works of authorship, and "performing," or engaging in live, real-time communication with others. Artifacts in turn are either static (with fixed content in a fixed order) or dynamic (programmed to present different content in different orders depending on external triggers (such as a user's behavior interacting with it). Performances fall into two high-level categories: those in which communication is unidirectional, or one-way, (such as speeches) and those in which communication is bidirectional (such as one-on-one and many-to-many conversations).

Some features apply to multiple branches of the Figure 1 tree:

- Most modes of expression can be through either physical or electronic means; for practical purposes, programmed content and social-media interaction can be accomplished only electronically;
- Electronically mediated expression can happen offline or online; that is, via electronic networks (such as the Internet) and protocols (such as the Web);
- Artifacts can include charts, diagrams, tables, flowcharts, decision trees, and other graphical elements; such things can also be used in most forms of performative expression;
- Artifacts can include audio and video elements that can also be used in performances; and
- Artifacts can include structural and navigational features (such as tables of contents, indices, and links); with physical artifacts the reader does the work; in electronic ones, buttons and hyperlinks make navigation easier. Many artifacts involve arbitrary access to any part (such as by page turning, fast-forwarding, and scene selection).

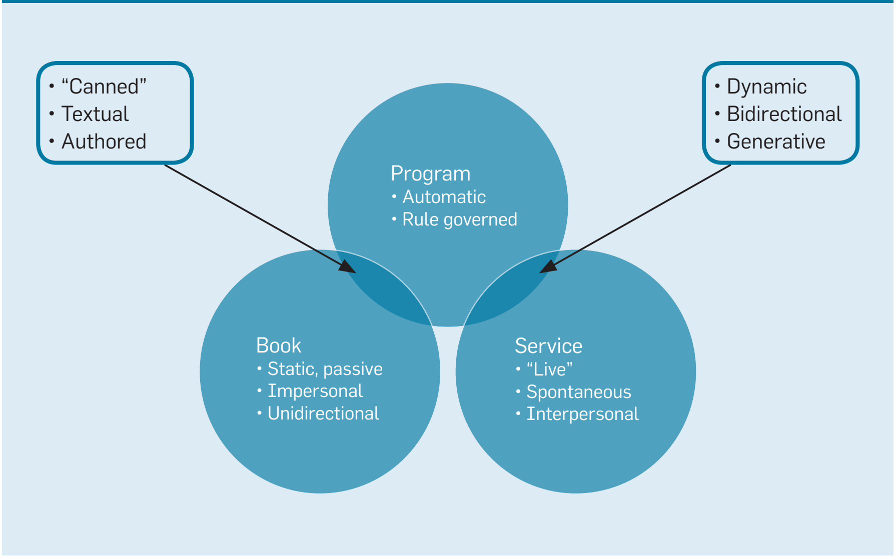
Are software programs more like books or like human services? The difficulty of reaching a satisfactory conclusion about automated legal assistance arises in part from our instinctive assent to two propositions:

- People should not be allowed to do through a program what they are not allowed to do in person; and
- People should not be disallowed to do through a program what they are allowed to do through books and other media.

To the extent a software application is viewed as a kind of personal conduct, it makes sense to apply the treatment one would apply to comparable functions being accomplished through an in-person service. To the extent a software application is viewed as a work of authorship, it makes sense to apply the treatment one would apply to the comparable content delivered through a book. How can these competing views be resolved?

We might first acknowledge that software applications are a tertium quid, or something similar to but distinct from both books and services.

Figure 2. Three modes of assistance.



p See *New York Lawyers Ass'n v. Dacey*, 234 N.E.2d 459 (N.Y. 1967)

q See, for example, Lanctot[4,5] and Oriola[9]

Like the wave/particle duality of light in modern physics, perhaps it makes sense to regard software as both a "work" and a service; Figure 2 outlines the shared and unshared characteristics of these kinds of things.

Software programs share characteristics with both books and instances of service delivery. Like books, they are essentially textual works of authorship, fully written in advance of their use; the author is not present at the time of use. Like services, they can be dynamic, bidirectional, and generative (such as by producing case-specific answers and documents). Unlike both, programs operate as machines, with automated behavior, and are rule-governed and deterministic.

Any of these modes of communication can be used for the transmission of knowledge, guidance, opinions, and expertise. The content being delivered can be "neutral" or tilted in favor of a particular kind of party or point of view.

Programs as texts. When in use, software applications typically involve no contemporaneous human involvement by their authors. Users interact with pre-written code, with no other human interacting with them as they do so.

Programs are special forms of words and numbers, textual objects that instruct machines how to behave. Any program can by definition be expressed textually. You can think of them, as hypertext pioneer Ted Nelson put it, as "literary machines."[8]

All outputs of an automated legal-assistance system are also in the form of textual speech acts. Delivering a document someone can download is not meaningfully different, except in terms of convenience, from presenting content that in effect says, "Here are the words you need, in this order."

That is, these systems not only emit texts, they *are* texts.

While debate among legal scholars continues as to whether the First Amendment extends to "symbolic" speech like flag burning,[10] there is little doubt it protects written texts. If I have the right to share the text of a program with others, and they would commit no offense by compiling and running it, why should I not have the right to run the program and give them access to it?

The question of whether First Amendment rights extend to computer code has arisen in cases involving publication of decryption algorithms; for example, "[C]omputer source code, though unintelligible to many, is the preferred method of communication among computer programmers. Because computer source code is an expressive means for the exchange of information and ideas about computer programming, we hold that it is protected by the First Amendment."[r]

Legality Broken

Like the world that inspired gamers in Jane McGonigal's 2011 book *Reality is Broken*,[7] the legal system in many countries is broken in many respects. Millions of people with pressing legal needs go without help. Courts are underfunded and overwhelmed. Many lawyers are unemployed or underemployed. Some law schools are struggling to survive. Recent law graduates are drowning in student loans.

Forbidding distribution of self-help legal software is not only of dubious wisdom as social policy, it is offensive to First Amendment values. It is difficult to make a principled case for suppressing freedom of expression about how the law works.

Free expression by definition need not be "authorized." Honest attempts to transmit knowledge about how the law works should not be suppressed, at least when done in ways that do not impersonate trusted lawyer/client relationships. Free citizens should not be required to have a license in order to express their understanding of how the law works or to sell or give away such expressions.

Coded law is not something, like hate speech at a military funeral, we should have to tolerate due to concern for higher values. It is an affirmative good we should embrace.

It is in the enlightened interest of lawyers, as well as the best interest of society in general, to enable programmatic expression of legal knowledge. We should be free to write code, run code, and let others run our code. If concerned citizens, law students, and entrepreneurs want to create tools that help people access and interact with the legal system, the government should not get in the way.

Are citizens at liberty to create and share software that helps others understand and interact with the legal system? Are we free to code the law?

We certainly should be.

r See *Junger v. Daley* 209 F.3d 481, 484-485 (6th Cir. 2000)

Acknowledgments
Research for this article was partially underwritten by LegalZoom.com. I am also grateful to Pamela Samuelson and Laurence Tribe for their helpful suggestions and reactions.

References
1. Dyson, G. *Turing's Cathedral: The Origins of the Digital Universe.* Pantheon Books, New York, 2012.
2. Grossman, M. and Cormack, G. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Richmond Journal of Law and Technology 17* (2011), 11–16.
3. Kimbro, S. *Virtual Law Practice.* American Bar Association, Chicago, 2010.
4. Lanctot, C. Scriveners in cyberspace: Online document preparation and the unauthorized practice of law. *Hofstra Law Review 30* (2002).
5. Lanctot, C. Does LegalZoom have First Amendment rights? Some thoughts about freedom of speech and the unauthorized practice of law. *Temple Political & Civil Rights Law Review 20* (2011).
6. Lauritsen, M. *The Lawyer's Guide to Working Smarter with Knowledge Tools.* American Bar Association, Chicago, 2010.
7. McGonigal, J. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World.* Penguin Press, New York, 2011.
8. Nelson, T. *Literary Machines: The Report On, and Of, Project Xanadu Concerning Word Processing, Electronic Publishing, Hypertext, Thinkertoys, Tomorrow's Intellectual Revolution, and Certain Other Topics, Including Knowledge, Education, and Freedom.* Mindful Press, Sausalito, CA, 1981.
9. Oriola, T. The use of legal software by non-lawyers and the perils of unauthorized practice of law charges in the United States. *Artificial Intelligence and Law 18*, 3 (2010), 285–309.
10. Volokh, E. Symbolic expression and the original meaning of the First Amendment. *Georgetown Law Journal 97* (2008).

Marc Lauritsen (marc@capstonepractice.com) is president of Capstone Practice Systems and Legal Systematics, Harvard, MA.

DOI:10.1145/2492007.2492026

How to understand evaluation criteria for CS researchers.

BY JACQUES WAINER, MICHAEL ECKMANN, SIOME GOLDENSTEIN, AND ANDERSON ROCHA

How Productivity and Impact Differ Across Computer Science Subareas

SOME COMPUTER SCIENCE researchers believe different subareas within CS follow different publishing practices, so applying a single production criterion would be unfair to some areas. It is reasonable to believe the subarea of, say, theory follows different publishing practices from subareas like software engineering and image processing. Scientific advances in theory are often bounded by the time needed to prove theorems. Moreover, at most institutions, CS faculty whose work involves theory advise fewer students and are thus likely to produce fewer publishable results per year.

» key insights

- We defined CS areas by selecting and combining people and publication venues.
- Journal productivity differs across CS areas, but differences in total productivity are less than we expected.
- The mean number of citations per paper varies depending on area.

It is also reasonable to believe CS subareas (we call them "CS areas" from here on) that deal mainly with data (such as image processing) are more likely to have greater productivity than areas in which evaluation procedures require users (such as human-computer interaction), programmers (such as software engineering), and organizations (such as management information systems). Researcher productivity in these human- and organization-based areas is bounded by the difficulty of carrying out the empirical evaluations the fields require. Though these beliefs are all reasonable, it is all they are, as they are as yet unproved.

Along with expected differences in productivity, we also often hear that different CS areas prefer and value conferences and journal publications in different ways; for example, bioinformatics seems more journal-oriented, while computer architecture seems more conference-oriented.

If there are indeed significant differences in publishing practices among the various CS areas, then a single production-based evaluation criterion for all CS researchers would favor some areas and disfavor others. A probable consequence, beyond possible unfairness to the disfavored areas, is that researchers would tend to avoid those areas in the future. Barbosa and Souza[1] discussed this problem with respect to a uniform publication evaluation standard in Brazil and its negative impact on human-computer interaction among Brazilian researchers.

Beyond publication practices, citation practices might also differ among areas. Areas with fewer researchers probably reflect fewer citations of papers published in these areas; a uniform evaluation-criteria impact of one's research across different CS areas would favor some areas while disfavoring others.

How to evaluate CS researchers has been discussed elsewhere, including Meyer et al.,[9] Patterson et al.,[10] and Wainer et al.,[14] emphasizing the differences in scientific publication culture between CS and other scientific domains; for example, Meyer et al.[9] discussed the importance of conference publication in CS, saying, a conference publication is, in some cases, more prestigious than a journal publication. The same general guideline of attributing importance to conferences is included in the Computing Research Association (CRA) guideline to faculty promotion in CS.[10] Wainer et al.[14] showed that a typical CS researcher's work is not represented in the standard citation services (such as Scopus and Thomson Reuters) compared to, say, mathematics and physics; thus, when using metrics based on these services, a CS researcher or university department could be unfairly evaluated, especially when competing against other disciplines. The role of conferences in CS has also been discussed by others; Grudin[6] collected many of the relevant articles and discussions, especially those published in *Communications*.

We are not aware of research that discusses the problems of uniform evaluation criteria across different CS areas, except for Barbosa and de Souza.[1] In other scientific disciplines (such as economics), some discussion focuses on the negative impact of uniform evaluation metrics on the different subareas of the discipline.[8]

General Description

Our methodology, as described here, relied on a sampling approach to evaluate the productivity and impact metrics of researchers and papers in different CS areas; we considered productivity as the number of articles published (in English) per year in journals, conferences, and workshops. Other than the distinction between journals and conferences (including workshops), we did not take into account any measures of venue quality (such as impact factor for journals and acceptance rate or scientific society sponsorships of conferences). The first step was to define the CS areas; the second to define the set of researchers working in each area, along with the set of conferences and journals associated with each area; the third to sample the set of researchers working in an area and collect from their own webpages the number of papers published from 2006 to 2010; and, finally, from the set of conferences and journals associated with a particular area, we sampled a set of papers and collected information about their citation counts. We briefly expand on each step; for a more detailed explanation of our methods, see Wainer et al.[15]

When deciding how to define and select a CS area, we were guided by some of the existing classification schemes. For example, ACM and IEEE each divides CS into different areas—ACM, through special interest groups, or SIGs, and IEEE, althrough technical committees, or TCs—though some of these divisions reflect historical decisions that may be less relevant today. DBLP, Microsoft Academic Search, and Scopus each classify different CS areas, though none describes how it arrived at its classifications.

We wanted our set of areas to include both new and more traditional CS areas, to evaluate whether or not the traditional areas follow publication practices that differ from the newer ones. Finally, we also wanted to include some areas on the "fringe" of CS that are not always present in university CS departments in different countries; Table 1 lists the areas we chose, the abbreviations we use for them, and the seed venues (using their usual abbreviations) for each area. We do not claim they are the only, or most important, areas of CS.

Bioinformatics and security are newer areas. Communications and networking, programming languages, databases, computer architecture, distributed computing, and software engineering are more traditional areas. Operations research and management information systems are the two "fringe" areas. Our choice of areas is compatible with, but not the same as, those of other research (such as Biryukov and Dong[2] and Laender et al.[7]) that also subdivides CS into areas.

For the second step, defining the population of researchers in each area and associated conferences and journals, we used DBLP data (as of August 2011) as our universe of interest. DBLP is a bibliographic server with a focus on CS that indexes more than 1.8 million CS articles.

Here, we use the words "publication venue" or just "venue" as a generic name for conferences or journals. We started by defining a set of venues clearly representative of each area, or "seed venues." The idea is that researchers in each area clearly recognize the seed venues as "central" and "important" to their area. We asked colleagues in each area and used information regarding citations received per published paper available by, say, Microsoft Academic Search and Thompson Reuters Journal of Citation Reports.

Using the set of venues associated with a particular area, we defined an iterative process that computes the set of researchers working in the area, then recomputes the set of venues in the area, and so on, until convergence. At the first iteration, the set of venues associated with an area is its seed venues, and the researchers working in the area are co-authors of at least two papers published in any seed venue from 2006 to 2010.

A new venue *v* is associated with area *A* if a clear majority of co-authors of papers in *v* (for the period 2006 to 2010) of researchers work in area *A*; for a formal definition of "clear majority," see Wainer et al.[15] Finally, all researchers publishing at least one paper in a seed venue and at least one in the newly added venue *v* are also considered researchers in area *A*. Note that a researcher may work in more than one area, but a venue is associated with at most one area. This method is a contribution of this research because it simultaneously classifies publications and authors by means of a semi-supervised algorithm, based on co-publication. Most publication-classification systems (such as Chen[3]) are based on unsupervised algorithms based on different citation links between journals and conferences. Other research (such as Rosen-Zvi et al.[12]) also uses unsupervised algorithms based on co-authorship and topic detection on the documents themselves.

At the end of the iterative process our algorithm revealed a universe of 56,589 researchers, of whom 4,827 work in more than one area. The algorithm also identified a set of 612 venues (247 journals and 365 conference proceedings); see Wainer et al.[15] for a list of all venues associated with each area.

The third step in our method was the sampling of researchers and papers. We randomly ordered the set of researchers in each area and sequentially searched each one until we found a set of 30 researchers with a personal or institutional webpage listing all their publications. We did not consider webpages that explicitly mentioned "selected" or "partial publication list"; 30 researchers per area was our attempt to balance the need to collect enough information about each area with the cost of finding researchers with a current list of publications.

Table 1. CS areas: names, abbreviations, and corresponding seed venues.

Area	Abbr.	Seed Publications
Artificial Intelligence	AI	AIJ, JAIR, JAR, AAAI, IJCAI
Bioinformatics	BIO	BMC Bioinf, Bioinformatics, JCB, RECOMB, TCBB
Communications and Networking	COMM	TON , TCOM, Mobicom , Sigcomm, Infocom
Compilers and Programming Languages	C+PL	OOPSLA, POPL, PLDI, TOPLAS, CGO
Computer Architecture	ARCH	ISCA , MICRO, DAC, ASPLOS, TCAD, SC
Computer Graphics	GRAPH	TOG, CGA, TVCG, SIGGRAPH
Database	DB	TODS, VLDB , Sigmod
Distributed Computing	DC	TPDS, JPDC, ICDCS, ICPP
Human-Computer Interaction	HCI	TOCHI, IJMMS, UMUAI, CHI, CSCW
Image Processing and Computer Vision	IPCV	IJCV, TIP, CVPR, ICIP
Machine Learning	ML	JMLR,ML, NECO, NIPS, ICML
Management Information Systems	MIS	ISR, MANSCI, JMIS, EJIS, MISQ
Multimedia	MM	MMS, TMM, IEEEMM, MM, ICMCS
Operational Research and Optimization	OR	Math Prog, SIOPT, C&OR, Disc Appl Math
Security	SEC	TISSEC, JCS, IEEESP, SP, USS, CSS
Software Engineering	SE	TOSEM, ICSE, TACAS, ESE
Theory	TH	JACM, SICOMP, STOC, FOCS, SODA

For each researcher, we collected the number of conference papers (including workshops but excluding posters) and journal papers listed on the researcher's page for the period 2006 to 2010 (inclusive) in English. We also collected the researchers' first and last publications years as listed on their webpages. The intersection of this interval with the interval 2006 to 2010 is a researcher's windowed publication interval. We defined the researcher's productivity as number of journal and conference papers published during that time divided by windowed publication interval. We also collected information on whether researchers were students (at the end of their windowed publication interval) and whether they were faculty in a non-CS university department. We considered non-CS any department that did not include the words "comput*" or "information" in its name.

For citation analysis, we randomly selected 100 papers for each area from all CS papers published in 2006 from all venues associated with a particular area (not just the seeds). In November 2011, we collected the number of citations received by each paper as compiled by Google Scholar. Given that citation counts are susceptible to outliers, we used the median number of citations per paper to perform the statistical calculations.

Finally, we used a 95% confidence level to make claims of statistical significance; see Wainer et al.[15] for the details of the statistical analysis.

Results

Table 2 lists some of the characteristics of the sample of 30 researchers per area. The column labeled "Stud." lists the number of students in the sample. The column labeled "Non-CS" lists how many researchers in the sample are faculty in non-CS departments. The column labeled "Resea." lists the total number of researchers in that area (based on the DBLP data), including those also working in other areas. The column labeled "Papers" refers to the total number of papers in all conferences and journals associated with the area published from 2006 to 2010.

The areas BIO, IPCV, MIS, and OR are the most "non-central" of the areas, based on the number of non-CS faculty in the sample. We were expecting non-centrality for BIO, MIS, and OR included in our set of areas specifically because they are not always represented in university CS departments; for BIO, some non-CS faculty were hosted in biology-related departments, for MIS, in business and marketing departments, and for OR in applied math and engineering departments. However, surprisingly, this pattern also holds for IPCV researchers, with about one-third affiliated with radiology and medical departments. The quantity of researchers in those areas not in CS departments may indicate those areas are more "interdisciplinary."[13]

The number of students in the sample may indicate, in a first approximation, the number of students working in each area. That number divided by the number of CS faculty in the sample may likewise be an indicator of the availability of students per CS researcher. Thus, MM and ARCH had the most students per CS researcher, while MIS, DC, and TH had the fewest students per CS researcher.

Productivity measures. Figure 1 outlines the mean conference and journal productivity of the sampled researchers in each area, in papers per year, ordered by total productivity (the sum of conference and journal productivity).

Table 3 lists the significant differences in total and journal productivity as a "compact letter display," a visualization tool showing nonsignificant differences between two areas; that is, the difference between the average total productivity of two areas is not statistically significant if the areas share a letter in common. When comparing any two areas, two or more letters in common means the same as one letter in common; that is, the areas are not significantly different. Only when two areas have no letters in common is the difference between them statistically significant. Each letter is an indicator of a maximal subset of areas in which differences are not statistically significant; for example, total production of DC ("d") is significantly different from that of DB ("ab"), since they have no common letter. However, DC ("d") is not significantly different from COMM ("cd") because they have the letter "d" in common. The second column lists the nonsignificant differences for journal productivity; for example, the OR in the journal productivity column includes all four letters ("abcd"), meaning OR is not significantly different from any other area, because OR has at least one letter in common with each other area.

Regarding total productivity, although the data seems to show three groups—higher productivity (ARCH, COMM, DC, and IPCV); middle; and lower productivity (MIS and OR)—almost all differences are not significant at 95% confidence level, as in Table 3. The only significant differences are, in general, between MIS and OR and the higher-productivity group. There are also significant differences between DB and TH and some of the higher-productivity areas but not all. Thus, one cannot claim that in general there are total productivity differences among the CS areas except for a few cases covered earlier.

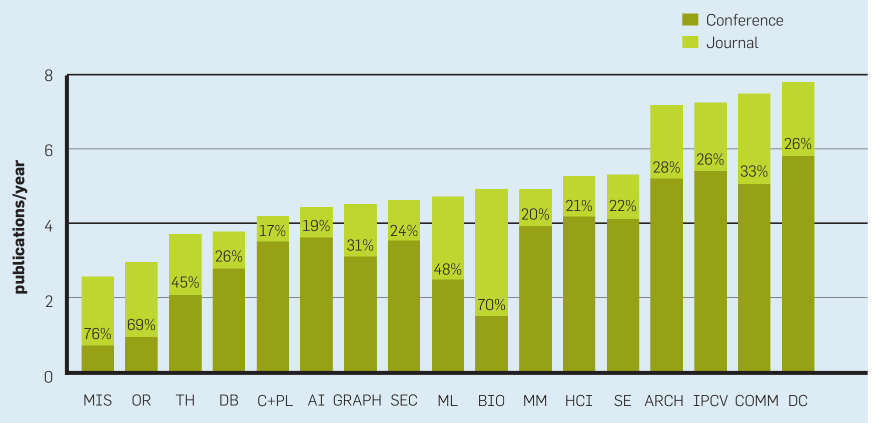
For journals, BIO has significantly higher productivity (3.44 papers per year) than all other areas, except the next four higher—COMM, MIS, ML, and OR; COMM is significantly different from the lower journal productivity areas—AI, C+PL, and DB—as in Table 3.

As for conferences, the highest two

Table 2. Characteristics of the samples and the population in each area: Stud. is number of students: Non-CS is number of non-CS faculty; CS is number of CS faculty in the sample; Resea. is total number of researchers; and Papers is total number of papers (published from 2006 to 2010) according to DBLP Computer Science Bibliography.

	Sample			Population	
Area	Stud.	Non-CS	CS	Resea.	Papers
AI	6	0	24	2244	4461
ARCH	10	3	17	3662	6666
BIO	4	11	15	8406	8037
C+PL	4	0	26	1001	1244
COMM	4	0	26	4395	6640
DB	8	0	23	1716	3066
DC	3	0	27	1112	1097
GRAPH	5	3	23	2176	2913
HCI	8	3	19	3229	5696
IPCV	4	12	12	6826	10959
MIS	0	19	11	1175	1800
ML	4	5	11	2619	3728
MM	9	0	11	3623	4790
OR	2	19	9	3103	6051
SE	4	2	24	2278	4993
SEC	4	0	26	1527	2690
TH	3	4	23	1595	5534

Figure 1. Production per year (in order of total productivity); the percent in each bar is the proportion of journal productivity to overall productivity.



conference productivity areas—DC and IPCV—are significantly different from the bottom third—MIS, OR, BIO, TH, ML, and DB—whereas the lowest two—MIS and OR—are significantly different from the top third—DC, IPCV, ARCH, COMM, HCI, and SE, information not in Table 3.

If we consider the ratio of journal productivity to total productivity, there are basically two groups: BIO, MIS, and, OR prefer journal publications to conferences, with about 70% of their production published in journals; the differences from all other areas are significant. ML and TH represent an intermediary group that publishes almost half its production in journals; the difference is statistically significant compared to most other areas.

Impact measures. Figure 2 includes the mean and median citations rates (citations per paper per year) for our sample of randomly selected 100 papers (from 2006) from each area (in order of median citations rate); the third column of Table 3 lists the compact letter display of the median citation rates for each area.

MIS citations rates are not significantly different from the next four higher rates—GRAPH, DB, BIO, and HCI—in decreasing order. The two lower-rate areas—ARCH and MM—are significantly different from the third lower-rate area—DC; the other areas are in the same group, with no significant differences among them.

The citation numbers reflect an interesting relation with productivity. The higher-productivity areas also have lower median citation rates. The correlation is moderately high and significant (Spearman rho = –0.63, p-value = 0.007). We use the Spearman rank correlation (rho) to detect any monotonic correlation between the variables, not just linear correlation. The correlation is even higher for conference productivity (rho = –0.71, p-value = 0.001). Thus, on the surface, in areas like ARCH and MM, researchers write many papers per year, especially conference papers, but few other researchers cite them. One notable aspect of the high-productivity/low-citation pattern is that the high-productivity areas tend to focus on conferences that, given the usual restrictions on number of pages in the publications, force authors to cite only a few relevant papers. However, a regression of the citation rates with both total productivity and proportion of journal publication reveals that only the negative coefficient of the total production is significant.

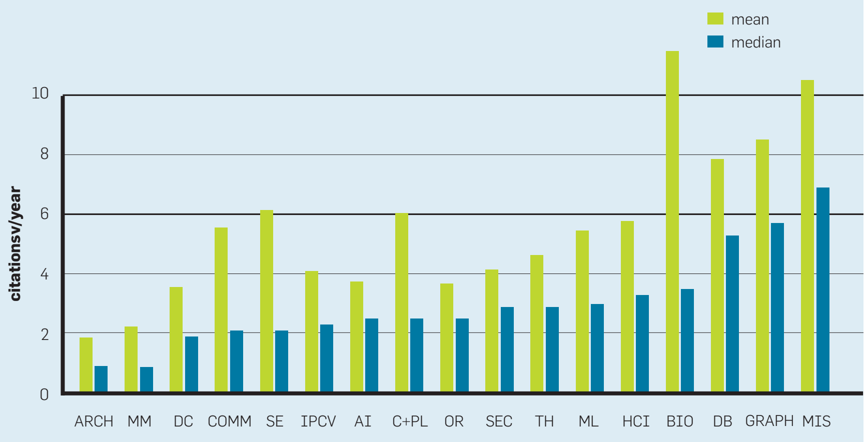
A reasonable hypothesis is thus that the median citation rate is correlated with the size of the area; that is, if an area includes few researchers, few potential readers are available to read the papers, inevitably yielding a low citation rate. However, the correlation between median citation rate and number of researchers in an area is not significant (rho = –0.26, p-value = 0.32) nor is the correlation with number of papers published in the area (rho = –0.25, p-value = 0.33). The size of a research area does not explain the different median citation rates.

The greater difference between mean and median citation rates for BIO, COMM, C+PL, and SE seem to indicate, at least in these areas, there is an even higher than usual concentration of citations on only a few papers, possibly increasing the mean but not the median. The two papers with the highest citation counts in our sample are from BIO.

Table 3. Compact letter display of total journal productivity and citations per paper per year; the difference between any two areas is not statistically significant if they have any letter in common.

Area	Average total productivity	Average journal productivity	Median citations per year
AI	abcd	ab	abc
ARCH	bcd	abc	de
BIO	abcd	d	ab f
C+PL	abcd	b	ab f
COMM	cd	cd	ab f
DB	ab	ab	fg
DC	d	abc	a c
GRAPH	abcd	abc	b fg
HCI	abcd	abc	abc
IPCV	bcd	abc	a c
MIS	a	abcd	g
ML	abcd	a cd	b fg
MM	abcd	abc	d
OR	a	abcd	c e
SE	abcd	abc	a c
SEC	abcd	abc	abc
TH	abc	abc	abc

Figure 2. Citations per paper per year (in order of median citation rate).



Discussion

Our productivity analysis found that although the productivity of the CS areas range from 2.5 (MIS) to 7.8 (DC) papers per year, the only significant differences are between the extremes of the spectrum. The total productivity of researchers in ARCH, COMM, DC, and IPCV is significantly higher than those for researchers in MIS and OR. The total productivity of the other areas does not differ significantly. Thus CS departments and evaluation bodies should be mindful when comparing researchers in MIS and OR to researchers in ARCH, COMM, DC, and IPCV.

Some evaluation criteria, especially those that apply to disciplines other than CS, put more emphasis on journal publications. CS departments that emphasize journal publications must be mindful that BIO in one group, and all marked areas without a "d" in the second column of Table 3 in the other, have significantly different journal productivity. However, BIO journal publication practices are not significantly different from those of COMM, MIS, ML, and OR.

There are more pronounced differences regarding whether the areas are conference- or journal-oriented in their publication practices. BIO, MIS, and OR are clearly journal-oriented and significantly different from the other areas. ML and TH are also significantly different from the most conference-oriented areas.

Regarding citations, there are significant differences among MIS (by itself), BIO, DB, HCI, and GRAPH (in another group), and finally, ARCH and MM. There is also an interesting negative correlation between productivity and citation rates beyond the influence of one area's emphasis on conference or journal publications.

Consider, too, these other interesting findings:

▸ We included BIO and SEC as examples of new CS areas. BIO indeed reflects very different publication and citation patterns from most other CS areas. SEC publication and citation patterns are not different from the majority;

▸ BIO, MIS, and OR are less-central CS areas, in the sense that a larger proportion of researchers in them are not in CS departments though, to our surprise, likewise IPCV. In some sense this non-centrality might indicate these areas are more interdisciplinary or multidisciplinary. In terms of publication and citation practices they differ somewhat from the bulk of CS, as discussed earlier, probably due to CS researchers adapting their practices to that of their research colleagues in other disciplines; and

▸ As far as our sampling was able to identify student availability per CS researcher, MM and ARCH seem to have the most students per CS researcher, while MIS, DC, and TH have the fewest.

CS areas that may be limited in their citation rates may consider encouraging all papers, especially conference papers, to include more elaborate analysis of the related literature.

Our research quantifies information researchers in the various CS areas already known, as in, say, the emphasis some of them put on conference publications. Some CS researchers have intuition regarding the differences among the areas derived from their personal observations of colleagues and acquaintances in these areas. However, as discussed earlier, before we began this research, this intuition should have been viewed as unproved beliefs gathered from a limited sample of convenience. We derive our conclusion from a random sample of 30 researchers worldwide and of 100 papers in each CS area. On the other hand, our research should be viewed as only a first step toward understanding the differences among CS areas. Moreover, our conclusions are limited by some issues that need to be further discussed:

The first is that our sampling of researchers introduced some bias. We discovered it is more likely that a non-senior faculty researcher in a university in a Western country would have an up-to-date publications page than the alternatives, including, say, a researcher in an Eastern country, students, industry-based researchers, and senior faculty researchers. Given that junior faculty are the researchers most likely to be evaluated through some of the metrics covered here, this bias has a limited effect. However, faculty in non-Western universities should take care when using our results, as they may not reflect their professional experience.

The second issue is sample size. Sampling researchers is labor intensive, so the sample size is small and the standard error associated with the measures is high. Not all differences are statistically significant at the level of 95%; some of our claims of nonsignificance may be revised if a larger sample is used; for example, we were surprised to find no statistically significant difference between the group of higher-productivity areas—ARCH, COMM, DC, and IPCV—and the middle group, including all areas but MIS and OR. A larger sample size might reveal whether this difference is significant.

Most research on bibliometric aspects of CS, including Biryukov and Dong,[2] Elmacioglu and Lee,[4] Franceschet,[5] and Laender et al.,[7] uses the whole DBLP as its data source. We could not follow such an approach. As pointed out by Laender et al.[7] and Feitz and Hoffmann,[11] DBLP has different coverage for different CS areas. If we used DBLP as the data source we would not know if the difference in productivity was due to the different practices of researchers in different areas or the difference in the DBLP coverage of these areas. We therefore used DBLP to define the populations and the set of researchers and publications in each CS area, but the final productivity measurements were not based on DBLP but on the papers listed on researchers' personal webpages. However, we used the DBLP data to define the size of each area in order to correlate it with the citation rates.

The procedure we describe here is repeatable. One may choose a different set of areas and initial seeds to explore more specific questions. The costly step is defining the sample, or finding which researchers have up-to-date webpages listing their publications. For help expanding on our results or exploring other issues regarding CS publication and citation practices, we provide the data used in this research at http://www.ic.unicamp.br/~wainer/datasets/CSareas/.

Our main purpose here is to provide the data needed to establish evaluation criteria for CS researchers; for example, one should not propose a single journal productivity goal for both COMM and AI researchers, as it would be unfair to AI researchers and could ultimately drive researchers away from the area. Productivity and citation rates differ between some but not all CS areas, and evaluation criteria for CS researchers must account for these differences.

This research does not answer why there are publication and citation differences between different CS areas and what might be done about them. We began by claiming there may be intrinsic differences among the areas and that doing research in one may be more difficult than in another; for instance, we mentioned that research in HCI, MIS, and SE could have lower productivity due to the difficulty of creating and conducting empirical research in these areas. However, HCI and SE have high productivity. The difference in total productivity of the three areas taken together when compared to the other areas, also taken together, is not statistically significant (t test p-value = 0.12). A second explanation for the differences in productivity is the availability of students. Again, the data we collected does not allow us to make that claim; the correlation between availability of students per CS researcher and total productivity is not significant (Spearman rho = 0.24 p-value = 0.36).

We also found another intrinsic possible explanation mentioned earlier to be false. There is no significant correlation between citation rates and the size of a CS research area. Surprisingly, all reasonable explanations for different productivity and citation rates across different areas mentioned here are not true as far as the data shows.

Conclusion

We are not able to claim one publication practice is "better" than another. Moreover, it may not be possible for a research community to change its publication practices without undergoing internal turmoil, though citation practices may be more amenable to change. Areas with low citation rates may look to areas like DB and GRAPH, which for most other characteristics are in the mainstream of CS practices but still have very high citation rates. It seems papers in these areas dedicate much more space to show how the research connects to previously published papers, with a corresponding increase in the references they include. CS areas that may be limited in their citation rates may consider encouraging all authors, especially of conference papers, to include more elaborate analysis and inclusion of the related literature. C

References

1. Barbosa, S.D.J. and de Souza, C.S. Are HCI researchers an endangered species in Brazil? *Interactions 18*, 3 (May/June 2011), 69–71.
2. Biryukov, M. and Dong, C. Analysis of computer science communities based on DBLP. In *Research and Advanced Technology for Digital Libraries, Volume 6273 of LNCS*. Springer, Berlin 2010, 228–235.
3. Chen, C. Classification of scientific networks using aggregated journal-journal citation relations in the journal citation reports. *Journal of the American Society for Information Science and Technology 59*, 14 (Dec. 2008), 2296–2304.
4. Elmacioglu, E. and Lee, D. On six degrees of separation in DBLP-DB and more. *ACM SIGMOD Record 34*, 2 (June 2005), 33–40.
5. Franceschet, M. Collaboration in computer science: A network science approach. *Journal of the American Society for Information Science and Technology 62*, 10 (Oct. 2011), 1992–2012.
6. Grudin, J. *A List of Publication Culture Discussions and Resources, 2012*; http://research.microsoft.com/en-us/um/people/jgrudin/CACMviews.pdf
7. Laender, A., de Lucena, C., Maldonado, J., de Souza e Silva, E., and Zivianim, N. Assessing the research and education quality of the top Brazilian computer science graduate programs. *ACM SIGCSE Bulletin 40*, 2 (June 2008), 135–145.
8. Lee, F., Grijalva, T., and Nowell, C. Ranking economics departments in a contested discipline: A bibliometric approach to quality equality between theoretically distinct subdisciplines. *American Journal of Economics and Sociology 69*, 5 (Nov. 2010), 1345–1375.
9. Meyer, B., Choppy, C., Staunstrup, J., and van Leeuwen, J. Research evaluation for computer science. *Commun. ACM 52*, 4 (Apr. 2009), 31–34.
10. Patterson, D., Snyder, L., and Ullman, J. *Evaluating Computer Scientists and Engineers for Promotion and Tenure*. Technical Report. Computing Research Association, Washington, D.C., Aug. 1999; http://www.cra.org/resources/bp-memos/evaluating_computer_scientists_and_engineers_for_promotion_and_tenure/
11. Reitz, F. and Hoffmann, O. An analysis of the evolving coverage of computer science subfields in the DBLP digital library. *Research and Advanced Technology for Digital Libraries. Volume 6273 of LNCS*. Springer, Berlin, 2010, 216–227.
12. Rosen-Zvi, M., Griffiths, T., Steyvers, M., and Smyth, P. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (Banff, Canada, July 7–11). AUAI Press, Arlington, VA, 2004, 487–494.
13. Wagner, C., Roessner, J., Bobb, K., Klein, J., Boyack, K., Keyton, J., Rafols, I., and Börner, K. Approaches to understanding and measuring interdisciplinary scientific research: A review of the literature. *Journal of Informetrics 5*, 1 (Jan. 2011), 14–26.
14. Wainer, J., Billa, C., and Goldenstein, S. Invisible work in standard bibliometric evaluation of computer science. *Commun. ACM 54*, 5 (May 2011), 141–146.
15. Wainer, J., Eckmann, M., Goldenstein, S., and Rocha, A. *Differences in Productivity and Impact Across the Different Computer Science Subareas. Technical Report IC-12-08.* Institute of Computing, University of Campinas, Campinas, Brazil, Mar. 2012; http://www.ic.unicamp.br/~reltech/2012/12-08.pdf

Jacques Wainer (wainer@ic.unicamp.br) is a professor in the Computing Institute of the University of Campinas, Campinas, Brazil.

Michael Eckmann (meckmann@skidmore.edu) is an associate professor in the Department of Mathematics and Computer Science of Skidmore College, Saratoga Springs, NY.

Siome Goldenstein (siome@ic.unicamp.br) is an associate professor in the Computing Institute of the University of Campinas, Campinas, Brazil.

Anderson Rocha (anderson@ic.unicamp.br) is an assistant professor at the Computing Institute at the University of Campinas, Campinas, Brazil.

DOI:10.1145/2493883

Taking a biologically inspired approach to the design of autonomous, adaptive machines.

BY JOSH C. BONGARD

Evolutionary Robotics

THE AUTOMATED DESIGN, construction, and deployment of autonomous and adaptive machines an open problem. Industrial robots are an example of autonomous yet nonadaptive machines: they execute the same sequence of actions repeatedly. Conversely, unmanned drones are an example of adaptive yet non-autonomous machines: they exhibit the adaptive capabilities of their remote human operators. To date, the only force known to be capable of producing fully autonomous as well as adaptive machines is biological evolution. In the field of evolutionary robotics,[9] one class of population-based metaheuristics—evolutionary algorithms—are used to optimize some or all aspects of an autonomous robot. The use of metaheuristics sets this subfield of robotics apart from the mainstream of robotics research, in which machine learning algorithms are used to optimize the control policy[a] of a robot. As in other branches of computer science the use of a metaheuristic algorithm has a cost and a benefit. The cost is that it is not possible to guarantee if (or when) an optimal control policy will be found for a given robot. The benefit is few assumptions must be made about the problem: evolutionary algorithms can improve both the parameters and the architecture of the robot's control policy, and even the shape of the robot itself.

Because the trial-and-error nature of evolutionary algorithms requires a large number of evaluations during optimization, in many evolutionary robotics experiments optimization is first carried out in simulation. Typically an evolutionary algorithm generates populations of virtual robots that behave within a physics-based simulation.[b] Each robot is then assigned a fitness value based on the quality of its behavior. Robots with low fitness are deleted while the robots that remain are copied and slightly modified in some random manner. The new robots are evaluated in the simulator and assigned a fitness, and this cycle is repeated until some predetermined time period has elapsed. The most-fit robot may then be manufactured as a physical machine and deployed to perform its evolved behavior.

To illustrate the distinction between mainstream and evolutionary robotics, consider two experiments drawn from the two fields. Legged locomotion—

a A control policy is some function that transforms a robot's sensor signals into commands sent to its motors.

b Interested readers may download and perform their own evolutionary robotics experiments at http://www.uvm.edu/˜ludobots.

» key insights

- Manual design of a mobile robot that is autonomous and adaptive is extremely difficult.
- As an alternative, computers can 'evolve' populations of robots in a simulator to exhibit useful behavior and then manufacture physical versions of the best ones, very much like how farmers breed crops for high yield. This approach is known as evolutionary robotics.
- This evolutionary approach changes the way we view robotics: rather than machine-learning techniques improving behaviors for a hand-designed robot, focus shifts to creating an evolutionary system that continuously designs and manufactures different robots with increasing abilities.

ILLUSTRATION BY JUSTIN METZ

optimizing a control policy that allows a two, four, or six-legged robot to move over rugged terrain—is a popular area of study in robotics. In mainstream robotics, machine-learning algorithms can now optimize walking behavior for a physical two-legged robot in a matter of minutes.[7] Alternatively, a recent investigation in simulation has shown if robots are evolved to move over rough terrain, robots will eventually evolve from amorphous shapes into robots exhibiting the rudiments of appendages (Figure 1b).[1]

The former experiment can enable walking behaviors for a certain kind of robot; the latter experiment can continuously produce different robots adapted to different environments. Put differently, mainstream robotics aims to continuously generate better behavior for a given robot, while the long-term goal of evolutionary robotics is to create general, robot-generating algorithms.

History

The goal of artificial intelligence, since its beginnings, has been to reproduce aspects of human intelligence (such as natural language processing or deductive reasoning) in computers. In contrast, most roboticists aim to generate noncognitive yet adaptive behavior in robots such as walking or object manipulation. Once these simpler behaviors are realized successfully in robots, it is hoped the behavior-generating algorithm will scale to generate ever more complex behavior until the adaptive behavior exhibited by a given robot might be characterized by an observer as intelligent behavior. This operational definition of intelligence bears a resemblance to the Turing Test: if a robot looks as if it is acting intelligently, then it is intelligent.

Note the emphasis in robotics on "behavior:" the action of a robot generates new sensory stimulation, which in turn affects its future actions. This differs from non-embodied AI algorithms, which have no body with which to affect, or be affected by the environment. In non-embodied AI, intelligence is something that arises out of introspection; in robotics, the belief is that intelligence will arise out of ever more complex interactions between the machine and its environment. This idea that intelligence is not just something contained within the brain of the animal or control policy of a robot but rather is something that emerges from the interaction between brain, body, and environment, is known as embodied cognition.[27]

The very first experiments in evolutionary robotics[9] began to shed light on embodied cognition. In one set of experiments a robot equipped with a camera had to move toward certain shapes and away from others. Based on the way the robot evolved to move, the control policy of the robot often only made use of two small pixel patches rather than the entire video stream. In other words, the robot evolved the ability to recognize objects through a combination of motion and sensation. This approach is non-intuitive to a human designer, who might implement object-recognition algorithms that draw on all of the pixels in the video stream.

Applications

Evolutionary algorithms have been applied in several branches of robotics and thus evolutionary robotics is not strictly a subfield of robotics. When applied well, an evolutionary approach can free the investigator from having to make decisions about every detail of the robot's design. In many cases the evolutionary algorithm discovers solutions the researcher might not have thought of, especially for robots that are non-intuitive for a human to control or design. For example it is often difficult to see how best to control a soft robot (Figure 1j) using traditional machine learning techniques, let alone determine the best combination of soft and rigid materials for such a robot.

Moreover, ideas can flow not just from biology to robotics but back again: evolved robots that exhibit traits observed in nature—such as a robot swarm that evolves cooperative rather than competitive tendencies—often provide new ways of thinking about how and why that trait evolved in biological populations. In this way evolutionary robotics can give back to biology ("Why did this trait evolve?") or more cognitively oriented fields such as evolutionary psychology ("Why did this cognitive ability evolve?").

Evolutionary biorobotics. In biorobotics, investigators implement anatomical details from a specific animal in hardware and then use the resulting robot as a physical model of the animal under study. Although much work in this area has been dedicated to nonhuman animals (see supplemental material available in the ACM Digital Library; http://dl.acm.org), many roboticists choose to model the human animal: a humanoid robot is more likely to be able to reach a doorknob, climb steps, or drive a vehicle than a wheeled robot or one measuring only a few inches in length. The humanoid form, however, requires mastery of bipedal locomotion, a notoriously difficult task. As an example, Reil et al.[30] evolved a bipedal robot in simulation that first mastered walking and then evolved the ability to walk toward a sound source.

In short, bioroboticists attempt to model, in robot form, the products of evolution: individual organisms. Evolutionary roboticists in contrast attempt to re-create the process of evolution, which generates robots that may or may not resemble existing animals.

Evolutionary biorobotics is a blend of these two approaches: investigators build robots that resemble a particular animal, and then evolve one aspect of the robot's anatomy to investigate how the corresponding aspect in the animal might have evolved. For example Long and his colleagues[19] have evolved the stiffness of artificial tails attached to swimming robots: robots with tails of differing stiffness have differing abilities to swim fast or turn well. This provides a unique experimental tool for investigating how backbones originally evolved in early vertebrates.

Developmental robotics. The field of developmental robotics[22] shares much in common with evolutionary robotics. Practitioners of developmental robotics draw inspiration from developmental psychology and developmental neuroscience: how do infants gradually mature into increasingly complex and capable adults? Like evolutionary robotics, work in developmental robotics tends to have either a scientific or an engineering aim. Developing robots can be used as scientific tools: they can serve as physical models for investigating biological development. Alternatively, engineers can draw on insights from biological development to build better robots.

Evo-devo-robo. Developmental robotics tends to focus on post-natal change to a robot's "body" and "brain" as the robot learns to master a particular skill. Evolutionary robotics experiments on the other hand generate robots that become more complex from generation to generation, but typically each individual robot maintains a fixed form while it behaves.

Biological systems however exhibit change over multiple time scales: individual organisms grow from infants into adults, and the developmental program that guides this change is in turn altered over evolutionary time. This process is known as the evolution of development, or *evo-devo*. This biological phenomenon has recently been exploited in evolutionary robotics:[3] At the outset of evolution, robots change from a crawling worm into a legged walking machine over their lifetime. As evolution proceeds, this infant form is gradually lost until, at the end of evolution, legged robots exhibit the ability to walk successfully without the need to crawl first. It was found this approach could evolve walking machines faster than a similar approach that does not lead robots through a crawling stage.

In the initial experiments of evo-devo-robo,[34] the genetic instructions were encoded as a specific class of formal grammars known as Lindenmayer systems, or L-systems.[c] L-systems were initially devised to model plant growth: their recursive nature can produce fractal or otherwise symmetric forms. Hornby[12] demonstrated that robots evolved using such grammars do indeed produce repeated forms (Figure 1a). He also showed this repetition can make it easier for evolutionary algorithms to improve such robots, compared to robots lacking in genetically determined self-similarity.

The evolution of robot bodies and brains differs markedly from all other approaches to robotics in that it does not presuppose the existence of a physical robot. Rather, the user provides as input a metric for measuring robot performance along with a simulation of the robot's task environment, and the algorithm produces as output the body plan and control policy for a robot capable of performing the task. This can then be used to manufacture a physical version of the evolved robot. Such an algorithm could, in principle, continually receive new desired behaviors and task environments and continuously generate novel robots.

The evolution of robot bodies and brains differs markedly from all other approaches to robotics in that it does not presuppose the existence of a physical robot.

In this way, the roboticist can make fewer assumptions about the final form of the robot and have greater confidence the final evolved robot is better adapted to the environment in which it must operate. For example, there is often a debate about whether a wheeled or legged robot is more appropriate for moving over a given surface. Although not yet demonstrated, an evolutionary robotics algorithm should generate wheeled robots if supplied with a simulation of flat terrain and legged robots if supplied with a simulation of rugged terrain. Recent work in mainstream robotics has demonstrated the possible advantage of combining wheels and legs in the same robot: an evolutionary system should rediscover this manually devised solution if it is indeed superior to either wheels or legs alone.

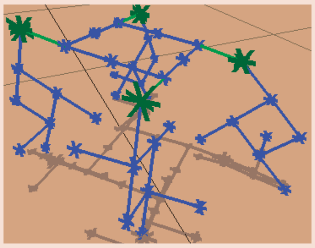
Another advantage of this approach over mainstream robotics is its potential for better scalability: by genetically encoding assembly instructions rather than the blueprint of a robot, more complex machines can be evolved with little or no increase in the amount of information encoded in the genome. For example, consider an approach in which robots are specified by a formal grammar such that the invocation of a rewrite rule replaces one part of the robot with two or more parts. Thus the more times a given set of rewrite rules are invoked, the more complex the resulting robot becomes. If evolution increases the number of rewrite rule invocations, then simple robots can evolve into more complex robots with no increase in the information content of the underlying genomes describing those robots.

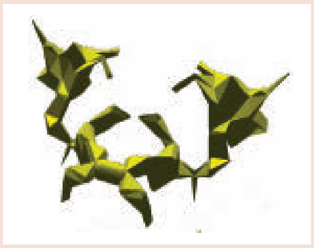
Despite the promise of this ap-

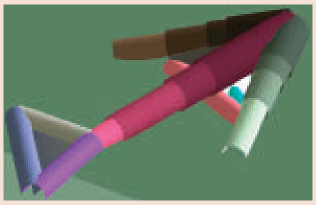
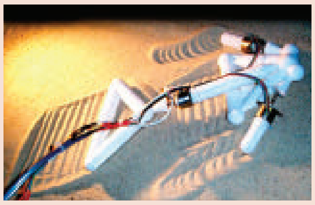
c Sims' work had a large impact on the computer graphics community and L-systems remain a popular technique within that field.

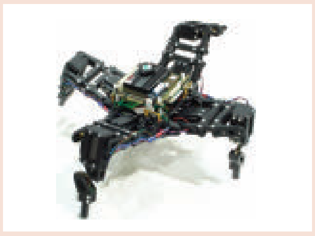
Figure 1. A sampling of representative work in evolutionary robotics.

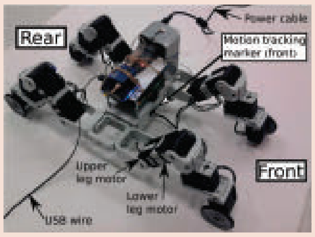
Evolutionary robotics often involves optimizing not only the controller of a robot but also its body plan. Formal grammars (a[12]) and algorithms that simulate development (b[1]) have been used to optimize robots in simulation. Additive manufacturing has been employed to build physical versions of evolved simulated robots semiautomatically (c,d[18]). Once deployed as physical machines, evolutionary algorithms have been used to allow damaged robots to recover from injury (e[4]) as well as ease the transferral of newly evolved controllers from simulation to the physical robot (f[16]). In addition to locomotion, researchers have evolved more cognitively demanding behaviors such as discriminating between differently shaped objects by manipulating them (g[35]) or physically demanding tasks like aerial swarming (h[11]). Behaviors have also been evolved for robots with non-traditional body plans such as tensegrity robots (i;[26] robot built by S. Fivat), soft robots (j[32]), modular robots (k[39]) and robot swarms (l[33]).


(a) Hornby and Pollack[12]

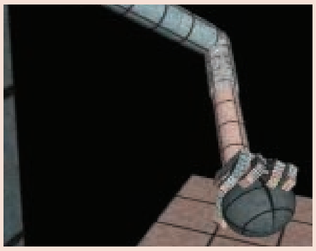

(b) Auberbach and Bongard[1]

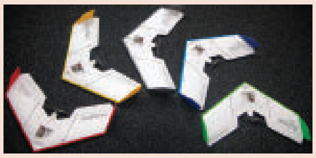


(c) The GOLEM Project Lipson and Pollack[18] (d)

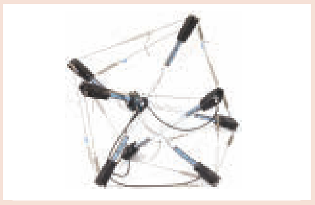

(e) Bongard et al.[4]

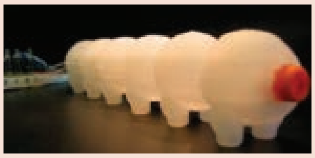


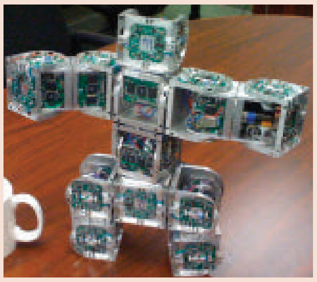
(f) Loos et al.[16]


(g) Tuci et al.[35]


(h) Hauert et al.[11]


(i) Paul et al.[2]


(j) Rieffel et al.[32]


(k) Yim et al.[39]
(USC Polymorphic Robotic Lab)

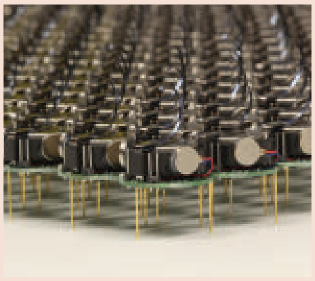

(l) Rubentstein et al.[33]

Figure 1c,d,f,g,j,k,l reprinted courtesy of IEEE.
Figure 1e reprinted courtesy of AAAS.
Figure 1h reprinted courtesy of Springer.

proach, only a handful of such algorithms have yet been developed. There are five main reasons for this. First, implementing such an algorithm is extremely difficult, as it requires a robust physics-based simulator that can accurately simulate complex mechanical constructs of arbitrary topology. Second, even with today's available computing power it can be computationally prohibitive to evaluate the thousands or millions of candidate robots required to generate one of sufficient quality. Third, an evolutionary algorithm must be devised that is expressive enough to encode diverse robot forms and evolvable in the sense that successive slight mutations lead to successively more complex and capable robots. Fourth, building a physical copy of the often complex virtual robots produced by such systems can be prohibitive. And finally, such systems have yet to automatically generate a robot that is more complex and capable than those designed and built manually. Overcoming these challenges remains a strong focus in the field.

Swarm robotics. One of the major challenges in swarm robotics is devising a control policy that, when executed by all members of the swarm, gives rise to some desired global behavior (Figure 1l). For example, if one wishes to program a group of robots to move collectively in a way similar to biological herds, flocks, or schools of fish, it has been shown[31] that each robot must balance attraction toward its local neighbors with repulsion away from neighbors that are too close.[d] However, if attraction is weighted too heavily, the swarm can contract into a traffic jam; if repulsion is weighted too strongly the group disperses.

This approach to controlling groups of robots is based on the principle of self-organization observed at many levels of biological systems: biological elements such as cells or organisms often form into cohesive patterns without a central control signal. This approach is desirable in robotics, in which communication limitations or the danger of failure may make designating one robot as the leader a difficult or risky proposition.

Evolutionary approaches have been used to optimize individual behaviors within a robot swarm. In the first such work,[29] control policies for homogeneous robots evolved that allowed them to move in concert, despite the lack of a leader. Repeatedly evaluating large numbers of candidate controllers on groups of physical robots places severe demands on the underlying hardware, so much work in this area has relied on simulations of robot swarms. This has enabled researchers to investigate more complex group behaviors, such as group hunting.[20] Such experiments require an understanding of co-evolution: one group's ability to overcome a second group makes it likely the second group will evolve to defend against the original group. This in turn exerts pressure for the original group to evolve a new strategy, and so on.

Co-evolution requires competition between groups, but also cooperation between individual group members. Evolutionary robotics has been used to investigate the conditions under which cooperation will arise, and how communication may evolve to support it. In an early study communication evolved in groups of "male" and "female" simulated robots so that female robots could call out to and attract males for mating.[36] It was observed that different dialects would evolve and compete with one another. More recent work with populations of simulated robots has demonstrated how distinct communication strategies can arise and that there are evolutionary advantages to more complex strategies.[38] These and other studies may provide unique tools for studying the evolution of biological communication strategies in general, and human language in particular. Such work could also provide a physical substrate on which to test hypotheses from game theory that involve deception, cooperation, and competition.

Modular robotics. Advancing technology has now made modular robotics feasible: Individual robots, or modules, may dynamically attach and detach from one another to create a robot with a constantly changing form (Figure 1k). It has been shown that evolutionary algorithms can be used to optimize behaviors for a modular robot in a fixed form (for example, see Zahadat[40]). More recently evolutionary methods have been used to enable modular robots to self-assemble from their constituent parts or reconfigure into different functional forms (for example, see Meng[23]). Continuously evolving novel forms and associated behaviors appropriate for a newly encountered environment remains an open problem in this area.

Soft robotics. With the exception of wheeled vehicles, robots are typically constructed from jointed collections of rigid parts, mirroring the skeletal linkages of higher animals and humans. Advances in materials science, however, have made non-traditional robot body plans possible. As one example, the evolution of behaviors for tensegrity robots was reported in Paul[26] (Figure 1i). Tensegrity structures are collections of rigid and elastic links attached in a particular way that provide several advantages over traditional robots, such as the ability to automatically revert to their default form if perturbed.

Soft robots are emerging as a new class of machine that combines discrete rigid parts with continuous, soft materials (Figure 1j). Such machines could "squeeze through holes, climb up walls, and flow around obstacles."[32] Controlling such devices is non-trivial, as motion at one location of the robot can propagate in unanticipated ways to other parts of the body. Despite this, Rieffel et al.[32] successfully evolved locomotion for a soft robot such that it exploited rather than fought against the synergies within its body. Evolving the architectures of such discrete and continuous devices demands new kinds of optimization methods. Coupled with the sudden recent interest in this field[13] there are many contributions that computer scientists interested in optimization could make in this area.

Formalisms

Evolutionary robotics is a mostly empirical endeavor, although three formalisms—the nature of computation, dynamical systems theory, and information theory—are beginning to

d This basic algorithm has since become the cornerstone of computer graphics algorithms which simulate the movement of animal or human groups.

provide a theoretical foundation for the field.

Morphological computation. As noted earlier, evolutionary robotics builds on the concept of embodied cognition, which holds that intelligent behavior arises out of interactions between brain, body, and environment.[27] An important corollary of embodied cognition is that, given the right body plan, a robot (or animal) can achieve a given task with less control complexity than another robot with an inappropriate body plan. For example, a soft robot hand can grip a complex object simply by enclosing it: the inner surface of the hand passively conforms to the object. A robot hand composed of hard material must carefully compute how to grasp the object. It has been argued that the physical aspect of a robot—its morphology—can actually perform computations that would otherwise have to be performed by the robot's control policy if situated in an unsuitable body plan. This phenomenon of morphological computation[25] cannot be completely abstracted away from the physical substrate that gives rise to it in the way a Turing Machine can. Practitioners in this area would greatly benefit from the aid of theoretical computer scientists to formalize this concept.

Dynamical systems theory. Dynamical systems theory is increasingly a useful tool for creating controllers for autonomous robots.[2] Often these controllers take the form of artificial neural networks that have their own intrinsic dynamics: they exhibit complex temporal patterns spontaneously. Evolutionary algorithms can then be used to shape the parameters of these networks such that they can be pushed by incoming sensor stimuli to fall into desired attractor states. For example, a neural network that falls into a periodic attractor may generate a rhythmic gait in a legged robot. However, it has been demonstrated that a one-to-one mapping between a basin of attraction in a neural network and a distinct robot behavior may be overly simplistic,[14] indicating there is much work to be done at the interface of dynamical systems theory and evolutionary robotics.

Information theory. Typically in an evolutionary robotics experiment, the "fitness" of a robot is measured based on its ability to perform a given behavior, such as how far it can walk or how well it can grasp an object. Surprisingly, it has been found that maximizing certain information-theoretic measures within the neural network of evolving robots can lead to useful behavior.[28] Why information maximization produces desired behaviors rather than useless, random, or uninteresting behavior remains mostly unresolved, although some progress has been made in this direction.[8]

In addition to helping with the synthesis of behavior, information theory can also be used to analyze evolved behaviors. Williams et al. have recently shown[37] that information flow—the transfer of information from one variable to another—can be employed to measure how behaving robots "offload" computed information to their body and/or their environment. This technique therefore holds promise for formalizing the concept of morphological computation.[25]

Challenges

There are a number of challenges currently facing the field, including transferring evolved robots from simulation to physical machines; scalability issues; and the difficulty of defining appropriate fitness functions for automatically measuring behavior.

The Reality Gap Problem. Both biological and artificial evolution are notorious for exploiting the potential relationship between the animal (or robot) and its environment to produce new behaviors. For instance the lightweight property of feathers, which are thought to have originally evolved for heat regulation, was later exploited for flight.[e]

As an example of the exploitative tendencies of evolutionary algorithms applied to robots, a robot was initially designed to brachiate along a suspended beam.[10] The robot was composed of a main body slung under two arms, and a heavy battery pack attached to the main body. Gradually, the evolutionary algorithm discovered control policies for the robot that exploited, rather than fought against the weight of the batteries. These control policies would cause the robot to move such that the battery pack swung forward under the robot's body before it changed hand holds. This would cause the robot's center of mass to move forward, thus requiring much less force to release contact with the beam and grasp it further forward. This mimics the way primates exploit the weight of their bodies like a pendulum to bring them into reach of a new tree limb. It is also reminiscent of the energy-saving passive dynamics of bipedal locomotion (see more details in the supplemental material).

However, if robots are optimized in a simulator, artificial evolution may exploit simplifications or inaccuracies in how physics is simulated. Such evolved control policies may then fail to reproduce the desired behavior when transferred from simulated to physical robots. For example, if there is no noise in the simulator, a control policy may evolve to generate behavior based on a very narrow range of sensor values. If this control policy is then transferred to a physical robot with a sensor that registers a wider range of values due to limitations in its electronics or mechanics, the physical robot may not behave as intended. This failure of evolved solutions to "cross the gap" from simulation to reality is known as the "reality gap" problem[15] and is one of the major challenges facing the field. However, a number of solutions have been proposed and significant progress is being made in this area.

In early work, sampling of the physical sensors was conducted and used to simulate the robot's sensors during evolution.[24] Alternatively, noise can be added to different aspects of the robot and its interaction with the environment: noise can be added to the sensors, to the effects of the motors, or the position of the robot itself.[15] This keeps evolution from exploiting artifacts of the simulation.

However, neither of these approaches scale well. If the robot must interact with increasingly complex and asymmetric objects, more samples must be taken from the sensors that detect the object: the sensor must be polled

e This tendency of evolution to repurpose traits is known as "exaptation."

at many more distances and positions relative to the object because there are more unique views of the object from the standpoint of the sensor. If noise is added to the simulation, each control policy must be evaluated several times such that controllers evolve to be robust to the noise in the simulation. For more complex robots, noise must be added to greater numbers and types of sensors and actuators. This requires even more evaluations to evolve robustness against this larger number of noise sources.

In more recent work the typically unidirectional approach of transferring evolved control policies from simulated to physical machines has been replaced with bidirectional approaches in which optimization alternates between simulation and reality.[4,16] For example, in Bongard et al.[4] three different evolutionary algorithms were employed. The first optimized a population of physical simulators to better reflect reality: The fitness of a simulator was defined as its ability to predict the behavior of the physical robot (Figure 1e).

The second evolutionary algorithm optimized exploratory behaviors for the physical machine to perform. These behaviors were assigned a high fitness if, when executed by the physical machine, they extracted the most new information about the way in which the robot could interact with its environment. This new information then became new training data for the first evolutionary algorithm. Gradually, after several alternations between these two optimization methods, a physical simulation would automatically emerge that was adapted to the details of the quadrupedal physical robot that was used in the experiment. The third evolutionary algorithm then uses this highly fit simulator to evolve control policies for the physical robot, and it was found that many such evolved behaviors transferred successfully from simulation to reality.

This approach turned out to have an added advantage over previous attempts to cross the reality gap: the robot could recover from physical damage such as the mechanical separation of one of its four legs. If the robot experienced such damage while behaving, the robot could not directly sense the damage but there would be an inevitable change in the incoming sensor values. This change would be automatically incorporated by the first evolutionary algorithm into new simulations: simulations of a three-legged robot would gradually replace simulations of a four-legged robot. These new simulations would then be used to evolve new control policies for the damaged robot that would allow it to automatically compensate for its injury.

Future work in this area would benefit from collaborations with developers of physical simulation such that evolution could alter the physical constants of the simulation itself, such as those used to model friction, collision, as well as aero- and hydrodynamics.

Koos et al.[16] recently proposed a different approach to the reality gap problem. Control policies evolved in simulation are transferred to a physical machine (Figure 1f), and the disparity between the behavior observed in simulation and reality is measured. This is done for several controllers, and the resulting disparity measures are used to create a model that predicts the disparity of control policies that have yet to be validated on the physical machine. A multi-objective optimization is then employed to maximize the desired behavior in simulation and to minimize predicted disparity: control policies are sought that generate the desired behavior in the simulated robot and are likely to reproduce that behavior in the physical robot.

This work attempted to address a seeming trade-off between behavioral efficiency and transferability: the more efficient the robot is at exhibiting a desired behavior the less likely it is to transfer to the physical machine. For example if fast-legged locomotion is selected for in simulation, running is more desirable than walking. However, running requires the robot's control policy to carefully manage its center of mass to avoid falling. If the mass distributions of the simulated and physical robot are slightly different, running behaviors may fail when transferred. This failure is less likely for walking behaviors in which the robot's mass distribution is less important.

It was identified early on that the time required to evaluate a single robot might grow exponentially with the number of parameters used to describe its task environment.

Most of the work on the reality gap problem has assumed that only the control policy of robots will be transferred. Lipson and Pollack,[18] however, integrated an evolutionary robotics simulation with rapid prototyping technology to automate robot manufacture as well as robot design (Figure 1c, 1d). They first evolved the body plans and control policies for robots composed of linked assemblages of linear actuators. Then, the 3D architectures of the best of these evolved robots were printed out of plastic; motors, circuitry, and batteries were then added by hand. Many of these automatically designed and manufactured robots were able to successfully reproduce the locomotion patterns originally evolved in the simulator.

Combinatorics of evaluation. It was identified early on that the time required to evaluate a single robot might grow exponentially with the number of parameters used to describe its task environment.[22] For example, consider a robot that must grasp m different objects under n different lighting conditions. Each robot must be evaluated for how well it grabs each object under each lighting condition, requiring mn evaluations per robot. If there are p parameters describing the task environment and each parameter has s different settings, then each robot must be evaluated s^p times.

This is a serious challenge in the field that has yet to be resolved. However, one possible solution to this challenge may be addressed using co-evolution. Consider a population of robots and a second population of task environments competing against one another. The robots evolve to succeed when exposed to environments drawn from the pool of evolving environments, and environments evolve to foil the abilities of the evolving robots. This is not unlike prey evolving to elude predators, while the predators evolve to catch prey. This approach could, in the future, be used to evolve robots that successfully generalize against a subset of task environments they might encounter when manufactured and deployed.

Evolvability. Evolving all aspects of a complex machine such as a robot is a daunting, high-dimensional optimization problem. Biological evolution faces the same challenge yet seems to have addressed it by a process known as the evolution of evolvability. A species with high evolvability is defined as one that can more rapidly adapt to changes in its environment than a similar species with lower evolvability.

One goal in evolutionary robotics in particular, and the field of evolutionary computation in general, is to create increasingly evolvable algorithms.

One goal in evolutionary robotics in particular, and the field of evolutionary computation in general, is to create increasingly evolvable algorithms. Rather than independently optimizing individual parameters of a candidate solution, such algorithms should rapidly discover useful aggregate patterns in candidate solutions and subsequently elaborate them. It has been shown, for example, that genomes that encode formal grammars produce robots with regular structure, and that such genomes are more evolvable than genomes that do not produce regular structures.[12]

Similarly, when an evolutionary algorithm biased toward producing regular patterns was used to evolve artificial neural networks for robots it was found, again, that such networks more rapidly discover desired behavior compared to other evolutionary methods that do not generate such regularity.[6] Auerbach and Bongard[1] have expanded the reach of this evolutionary algorithm to shape robot body plans as well.

Despite these recent advances, little is known about how to design evolutionary algorithms that reorganize genetic representations to maximize evolvability and thus automatically generate adaptive complex machines in a reasonable amount of time.

Fitness Function Design. The original and continued goal of evolutionary robotics is to make as few assumptions about the final form of the robot or the kind of behavior that should be generated. However, designing a fitness function that rapidly discovers desirable solutions without biasing it toward particular solutions is notoriously difficult. For this reason there have been efforts in the field to eliminate the usage of a fitness function altogether. One recent example is novelty search, which begins with simple candidate solutions and gradually creates more complex solutions as optimization proceeds.[17] The fitness of any given solution is simply how much it differs from previously generated solutions. This approach was found to produce walking in simulated bipedal robots, a notoriously difficult problem in robotics.

Conclusion

Since its founding in the early 1990s, evolutionary robotics has remained a small but productive niche field. Although the field has yet to evolve a robot that is superior to one produced using mainstream optimization methods such as reinforcement learning, the field has produced a wider variety of robots automatically. Depending on how one counts, roboticists have manually designed and built a few hundred different kinds of robots with humanoid or legged or snakelike body plans. Evolutionary methods, on the other hand have produced millions of different kinds of robots that can walk (for example, Figure 1a–d), swim, or grasp objects.[33] It is hoped that by exploring all the different ways that robots achieve these basic competencies we might gain unique insight into how to scale robots up to perform more complex tasks, like working safely alongside a human.

Moreover, several recent advances in fields outside of robotics are providing opportunities to showcase the advantages of this evolutionary approach. Advances in materials science are making soft robots and modular robots a reality, yet manually designing and controlling such robots is much less intuitive than traditional rigid and monolithic robots. Advances in automated fabrication are bringing the possibility of continuous and automated design, manufacture, and deployment of robots within reach. State-of-the-art evolutionary algorithms and physical simulators are making it possible to optimize all aspects of a robot's body plan and control policy simultaneously in a reasonable time period. And finally, new insights from evolutionary biology and neuroscience are informing our ability to create increasingly complex, autonomous, and adaptive machines.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under grant PECASE-0953837, and by the Defense Advanced Research Projects Agency (DARPA) under grants W911NF-11-1-0076 and FA8650-11-1-7155. C

References

1. Auerbach, J.E. and Bongard, J.C. On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the 2012 Genetic and Evolutionary Computation Conference*, 521–528.
2. Beer, R.D. The dynamics of brain-body-environment systems: A status report. *Handbook of Cognitive Science: An Embodied Approach* (2008), 99–120.
3. Bongard, J. Morphological change in machines accelerates the evolution of robust behavior. In *Proceedings of the National Academy of Sciences 108*, 4 (2011), 1234.
4. Bongard, J. Zykov, V. and Lipson, H. Resilient machines through continuous self-modeling. *Science 314* (2006), 1118–1121.
5. Cheney, N., MacCurdy, R., Clune, J. and Lipson, H. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, NY, 2013.
6. Clune, J., Beckmann, B.E., Ofria, C. and R.T. Pennock, R.T. Evolving coordinated quadruped gaits with the hyperneat generative encoding. *IEEE Congress on Evolutionary Computation* (2009), 2764–2771.
7. Collins, S., Ruina, A., Tedrake, R. and Wisse, M. Efficient bipedal robots based on passive-dynamic walkers. *Science 307*, 5712 (2005), 1082–1085.
8. Edlund, J.A., Chaumont, N., Hintze, A., Koch, C., Tononi, G. and Adami, C. Integrated information increases with fitness in the evolution of animats. *PLoS Computational Biology 7*, 10 (2011).
9. Floreano, D. and Mattiussi, C. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies.* MIT Press, Cambridge, MA, 2008.
10. Frutiger, D.R., Bongard, J.C. and Iida, F. Iterative product engineering: Evolutionary robot design. In *Proceedings of the Fifth International Conference on Climbing and Walking Robots*. P. Bidaud and F.B. Amar, eds. Professional Engineering Publishing, 2002, 619–629.
11. Hauert, S., Zufferey, J.C. and Floreano, D. Evolved swarming without positioning information: An application in aerial communication relay. *Autonomous Robotics 26* (2009), 21–32.
12. Hornby, G.S. and Pollack, J.B. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life 8*, 3 (2002), 223–246.
13. Iida, F. and Laschi, C. Soft robotics: Challenges and perspectives. *Procedia Computer Science 7* (2011), 99–102.
14. Izquierdo, E. and Buhrmann, T. Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: Walking and chemotaxis. *Artificial Life XI: Proceedings of the 11th International Conference on the Simulation and Synthesis of Living Systems*. MIT Press, Cambridge, MA, 2008, 257–264.
15. Jakobi, N., Husbands, P. and Harvey, I. Noise and the reality gap: The use of simulation in evolutionary robotics. *Advances in Artificial Life* (1995), 704–720.
16. Koos, S., Mouret, J.-M. and S. Doncieux, S. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation* (2012); doi: 10.1109/TEVC.2012.2185849.
17. Lehman, J. and Stanley, K.O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation 19*, 2 (2011), 189–223.
18. Lipson, H. and Pollack, J.B. Automatic design and manufacture of artificial lifeforms. *Nature 406* (2000), 974–978.
19. Long, J. *Darwin's Devices: What Evolving Robots Can Teach Us about the History of Life and the Future of Technology.* Basic Books, 2012.
20. Luke, S. and Spector, L. Evolving teamwork and coordination with genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming*. MIT Press, Cambridge, MA, 150–156.
21. Lungarella, M., Metta, G., Pfeifer, R. and Sandini, G. Developmental robotics: A survey. *Connection Science 15*, 4 (2003), 151–190.
22. Mataric, M. and Cliff, D. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems 19* (1996), 67–84.
23. Meng, Y., Zhang, Y. and Jin, Y. Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model. *Computational Intelligence Magazine 6*, 1 (2011). IEEE, 43–54.
24. Miglino, O., Lund, H.H. and S. Nolfi, S. Evolving mobile robots in simulated and real environments. *Artificial Life 2*, 4 (1995), 417–434.
25. Paul, C. Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems 54*, 8 (2006), 619–630.
26. Paul, C., Valero-Cuevas, F.J. and Lipson, H. Design and control of tensegrity robots for locomotion. *IEEE Transactions on Robotics 22*, 5 (2006), 944–957.
27. Pfeifer, R. and Bongard, J. How the Body Shapes the Way We Think: A New View of Intelligence. MIT Press, Cambridge, MA, 2006.
28. Polani, D., Sporns, O. and Lungarella, M. How information and embodiment shape intelligent information processing. In *50 Years of Artificial Intelligence*, Springer, 2007, 99–111.
29. Quinn, M. Smith, L., Mayley, G. and Husbands, P. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 361*, 1811 (2003), 2321–2343.
30. Reil, Y. and Husbands, P. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 159–168.
31. Reynolds, C.W. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics 21* (1987), 25–34.
32. Rieffel, J., Saunders, F., Nadimpalli, S., Zhou, H., Hassoun, S., Rife, J. and Trimmer, B. Evolving soft robotic locomotion in PhysX. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*. ACM, NY, 2009, 2499–2504.
33. Rubenstein, M., Ahler, C. and Nagpal, R. Kilobot: A low-cost scalable robot system for collective behanviors. In *Proceedings of 2012 IEEE International Conference on Robotics and Automation*. IEEE, 3293–3298.
34. Sims, K. Evolving 3D morphology and behaviour by competition. *Artificial Life*. Rodney A. Brooks and Pattie Maes, eds, (2009), 28–39.
35. Tuci, E., Massera, G., and Nolfi, S. Active categorical perception of object shapes in a simulated anthropomorphic robotic arm. *IEEE Transactions on Evolutionary Computation 14*, 6 (2010), 885–899.
36. Werner, G.M. and Dyer, M.G. Evolution of communication in artificial organisms. In *Proceedings of the Second International Conference of Artificial Life*. D. Farmer, C. Langton, S. Rasmussen, and C. Taylor, eds, (1991), 659–687.
37. Williams, P. and Beer, R. Information dynamics of evolved agents. In *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior*. S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, and J-B. Mouret, eds. Springer, 2010, 38–49.
38. Wischmann, S., Floreano, D. and Keller, L. Historical contingency affects signaling strategies and competitive abilities in evolving populations of simulated robots. In *Proceedings of the National Academy of Sciences 109*, 3 (2012), 864–868.
39. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E. and Chirikjian, G.S. Modular self-reconfigurable robot systems (grand challenges of robotics). *Robotics & Automation Magazine 14*, 1 (2007). IEEE, 43–52.
40. Zahadat, P., Christensen, D., Schultz, U., Katebi, S. and Stoy, K. Fractal gene regulatory networks for robust locomotion control of modular robots. In *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior*. S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, and J-B. Mouret, Eds. Springer, 2010, 544–554.

Josh C. Bongard (josh.bongard@uvm.edu) is director of the Morphology, Evolution and Cognition Laboratory in the Department of Computer Science at the University of Vermont. He is also a member of the Vermont Advanced Computing Core and the Complex Systems Center.

P. 86

Technical Perspective
Every Graph Is Essentially Sparse

By Assaf Naor

P. 87

Spectral Sparsification of Graphs: Theory and Algorithms

By Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng

DOI:10.1145/2492007.2492028

Technical Perspective Every Graph Is Essentially Sparse

By Assaf Naor

THE FOLLOWING PAPER by Batson, Spielman, Srivastava, and Teng surveys one of the most important recent intellectual achievements of theoretical computer science, demonstrating that *every* weighted graph is close to a sparse graph. As is often the case in key mathematical discoveries, a major part of the new contribution is a definition rather than just a theorem: here the authors describe an appropriate notion of "closeness" between graphs, called *spectral similarity*. This notion is fine enough so that graphs that are close to each other share certain properties which are crucial for a variety of algorithmic tasks, yet at the same time it can be argued that every graph is close to a graph which has few edges.

In the present (very general) context, an n-vertex weighted graph is nothing more than an n by n symmetric matrix $G = (g_{ij})$ consisting of nonnegative real entries, that is, $g_{ij} = g_{ji} \geq 0$ for every $i, j \in \{1, \dots, n\}$. The underlying combinatorial graph induced by G corresponds to its *support*: simply draw an edge joining a pair of vertices $i, j \in \{1, \dots, n\}$ if and only if the entry g_{ij} is strictly positive. This is a general template for modeling pairwise interactions between n items, where the quantity g_{ij} measures the intensity of the interaction between the ith item and the jth item.

A new structural paradigm for graphs would potentially impact numerous areas of research, since clearly graphs permeate computer science, mathematics, and the sciences in general. Therefore, at this level of generality, can it be possibly true that not all the important structural results for weighted graphs have already been discovered decades ago? The authors demonstrate through their remarkable work over the past years that the answer to this question is a resounding "yes."

Consider a graph G as a template for computing a certain "energy": whenever one assigns a real value x_i to each vertex i, the graph outputs the quantity $\frac{1}{2}\sum_{i,j=1}^{n} g_{ij}(x_i - x_j)^2$. As one ranges over all possible choices of real numbers $x_1, \dots, x_n$, this quantity encodes a lot of useful information about the structure of the graph G. For example, by restricting to the special case when the x_i are either 0 or 1, one recovers the entire *cut structure* of the graph, that is, for every subset S of the vertices, this quantity is nothing more than the total weight of the edges joining S and its complement (the size of the *boundary* of S in the graph G).

Given a small positive number ε, two graphs $G = (g_{ij})$ and $H = (h_{ij})$ on the same vertex set $\{1, \dots, n\}$ are said to be $(1 + \varepsilon)$-spectrally similar if the ratio between $\sum_{i,j=1}^{n} g_{ij}(x_i - x_j)^2$ and $\sum_{i,j=1}^{n} h_{ij}(x_i - x_j)^2$ is between $1 - \varepsilon$ and $1 + \varepsilon$ for *all choices* of real numbers $x_1, \dots, x_n$.

The multiyear effort of the authors yielded the following beautiful discovery: every graph G is $(1 + \varepsilon)$-spectrally similar to a graph H with at most $4\varepsilon^{-2}n$ edges. Thus, the matrix H reproduces the quantities $\sum_{i,j=1}^{n} g_{ij}(x_i - x_j)^2$ with high accuracy, and at the same time the average number of nonzero entries per row of H is bounded by a quantity that does not grow with n. The proof of this fact is constructive, yielding a deterministic polynomial time algorithm that takes G as input and outputs its sparse approximation H. For certain applications, one requires faster algorithms whose running time is nearly linear, and the authors show how to obtain such algorithms if one uses randomization and allows for output graphs H that have average degree that grows polylogarithmically with n.

Many applications of these results have already been discovered, and surely more will be found in years to come. The authors present some of these applications in the following article, including the design of a remarkable nearly linear time algorithm for approximate solution of diagonally dominant symmetric linear systems. Since the authors focus on applications to algorithm design, they do not describe the variety of mathematical applications of their work that have been discovered in recent years. For example, Barvinok recently used their work to approximate every centrally symmetric convex body by a polytope with few vertices.

The ideas and results detailed by Batson, Spielman, Srivastava, and Teng are important due to their generality and wide applicability. But they are a major intellectual achievement mainly due to the new ideas that are introduced in the *proofs* of their results. While related questions have been studied by mathematicians and computer scientists before, the method that the authors use is highly original and manifestly different from previous approaches. Their strategy to solve such questions introduced a new paradigm, and indeed rather than only using their results as a "black box," striking applications have been found by delving into their proof technique and adapting it to other problems (for example, this leads to a very original new approach to the important restricted invertibilty principle of Bourgain and Tzafriri, and yields new results on random matrices).

The work presented here exemplifies the best of modern research on theoretical computer science, as it reshapes our mathematical understanding and in tandem leads to the design of fast new algorithms. C

Assaf Naor (naor@cims.nyu.edu) is a professor of mathematics in the Courant Institute of Mathematical Sciences at New York University.

DOI:10.1145/2492007.2492029

Spectral Sparsification of Graphs: Theory and Algorithms

By Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng

Abstract

Graph sparsification is the approximation of an arbitrary graph by a sparse graph.

We explain what it means for one graph to be a spectral approximation of another and review the development of algorithms for spectral sparsification. In addition to being an interesting concept, spectral sparsification has been an important tool in the design of nearly linear-time algorithms for solving systems of linear equations in symmetric, diagonally dominant matrices. The fast solution of these linear systems has already led to breakthrough results in combinatorial optimization, including a faster algorithm for finding approximate maximum flows and minimum cuts in an undirected network.

1. INTRODUCTION

A sparse graph is one whose number of edges is reasonably viewed as being proportional to its number of vertices. In contrast, the complete graph on n vertices, with about $n^2/2$ edges, is the paradigmatic dense graph. Sparse graphs are often easier to handle than dense ones. Most graph algorithms run faster, sometimes by orders of magnitude, when there are fewer edges, and the graph itself can be stored more compactly. By approximating a dense graph of interest by a suitable sparse one, one can save time and space.

We will work with weighted graphs, where the weights might represent capacities, conductance, similarity, or just coefficients in a linear system. In a sparse graph, all of the edges can be important for a graph's structure. In a tree, for example, each edge provides the only path between its endpoints. Not so in a dense graph, where some edges will serve similar functions. A collection of low-weight edges connecting two clusters of vertices in a graph might be approximable by a single high-weight edge connecting vertices representative of those clusters. Sparsification can be viewed as a procedure for finding a set of representative edges and weighting them appropriately.

What exactly do we mean by sparse? We would certainly consider a graph sparse if its average degree were less than 10, and we would probably consider a graph sparse if it had one billion vertices and average degree one hundred. We formalize the notion of sparsity in the usual analysis-of-algorithms way by considering infinite families of graphs, and proclaiming sparse those whose average degrees are bounded by some constant, or perhaps by a polynomial in the logarithm of their number of vertices.

One may at first think that sparsification is unnecessary, as common wisdom holds that all large real-world graphs are sparse. While this may be true of natural graphs such as social networks, it is not true of the graphs that arise inside algorithms. Many algorithms involve the construction of graphs that are dense, even when solving problems on graphs that are sparse. Moreover, the common wisdom may be an artifact of the difficulty of storing and manipulating a large dense graph. Improvements in sparsification may one day ameliorate these difficulties.

The use of sparse graphs to approximate dense ones is not unique to algorithm design. In a parallel computer, for instance, information needs to be able to flow from any processor to any other. Hardwiring all those pairwise connections would be physically difficult, so a sparse graph simulating the connectivity properties of the complete graph is needed. The hypercube graph plays this role in the CM5, built by Thinking Machines.[25] Intuitively, the hypercube has no "bottlenecks." Formally, the (weighted) hypercube is a good spectral sparsifier for the complete graph defined on its nodes. We have shown that *every* graph has a very sparse spectral approximation, with constant average degree.

2. NOTIONS OF SIMILARITY

A few conventions: we specify a weighted graph by a 3-tuple, $G = (V, E, w)$, with vertex set $V = \{1, \ldots, n\}$, edge set $E \subseteq \{(u, v) \mid u, v \in V\}$, and weights $w_{(u,v)} > 0$ for each $(u, v) \in E$. All graphs will be undirected and weighted, unless otherwise stated. We sometimes express a graph simply by $G = (V, w)$, as E can be defined implicitly by setting $w_{(u,v)} = 0$ for all $(u, v) \notin E$. We will always write n for the number of vertices in a graph and m for the number of edges. When measuring the similarity between two graphs, we will always assume that they have the same set of vertices.

2.1. Cut similarity

The notion of cut similarity of graphs was first considered by Benczúr and Karger[8] as part of an effort to develop fast algorithms for the minimum cut and maximum flow problems. In these problems, one is interested in the sum of the weights of edges that are cut when one divides the vertices of a graph into two pieces. Two weighted graphs on the same vertex set are cut-similar if the sum of the weights of the edges cut is approximately the same in each such division.

To write this symbolically, we first observe that a division of the vertices into two parts can be specified by identifying

A previous version of the paper, "Twice-Ramanujan Sparsifiers," was published in the *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing* (2009), 255–262.

the subset of vertices in one part. For a weighted graph $G = (V, w)$ and a subset of vertices $S \subset V$, we define

$$cut_G(S) \overset{\text{def}}{=} \sum_{u \in S,\, v \in V-S} w_{(u,\,v)}.$$

We say that $G = (V, w)$ and $\tilde{G} = (V, \tilde{w})$ are *σ-cut similar* if

$$cut_{\tilde{G}}(S) / \sigma \le cut_G(S) \le \sigma \cdot cut_{\tilde{G}}(S),$$

for all $S \subset V$. Surprisingly, every graph is cut-similar to a graph with average degree $O(\log n)$, and that graph can be computed in polylogarithmic time.

THEOREM 1 (BENCZÚR-KARGER). *For all $\epsilon > 0$, every $G = (V, E, w)$ has a $(1 + \epsilon)$-cut similar graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$ such that $\tilde{E} \subseteq E$ and $|\tilde{E}| = O(n \log n/\epsilon^2)$. Moreover $\tilde{G}$ can be computed in $O(m \log^3 n + m \log n/\epsilon^2)$ time.*

The sizes of cuts in a graph tell us a lot about its structure—for starters, the weighted degrees of vertices are given by cuts of size $|S| = 1$. Most ways of defining a cluster of vertices in a graph involve comparing the number of edges in the cut defined by the set of vertices to the number of edges internal to that set.

2.2. Spectral similarity

Motivated by problems in numerical linear algebra and spectral graph theory, Spielman and Teng[34] introduced a notion of *spectral similarity* for two graphs. We will first describe it as a generalization of cut similarity.

Given a weighted graph $G = (V, w)$, we define the Laplacian quadratic form of G to be the function Q_G from $\mathbb{R}^V$ to $\mathbb{R}$ given by

$$Q_G(x) = \sum_{(u,\,v) \in E} w_{(u,\,v)} \left(x(u) - x(v)\right)^2.$$

If S is a set of vertices and x is the characteristic vector of S (1 inside S and 0 outside), then it is easy to see that

$$Q_G(x) = cut_G(S).$$

We say two graphs $G = (V, w)$ and $\tilde{G} = (V, \tilde{w})$ are *σ-spectrally similar* if

$$Q_{\tilde{G}}(x)/\sigma \le Q_G(x) \le \sigma \cdot Q_{\tilde{G}}(x), \quad \text{for all } x \in \mathbb{R}^V. \tag{1}$$

Thus, cut similarity can be viewed as the special case of spectral similarity in which we only consider vectors x that take values in $\{0, 1\}$.

It is possible to construct graphs that have very similar cuts, but which are highly dissimilar from the spectral perspective; for instance, the n-vertex path is 2-cut similar but only n-spectrally similar to the n-vertex cycle. Although spectral similarity is strictly stronger than cut similarity, it is easier to check if two graphs are spectrally similar. In particular, one can estimate the spectral similarity of two graphs to precision ϵ in time polynomial in n and $\log(1/\epsilon)$, but it is NP-hard to approximately compute the cut-similarity of two graphs.

Graphs that are spectrally similar share many algebraic properties. For example, the effective resistance distances between all pairs of vertices are similar in spectrally similar graphs. The effective resistance distance is defined by viewing each edge in a graph as a resistor: an edge of weight w becomes a resistor of resistance $1/w$. The entire graph is then viewed as a resistive circuit, and the effective resistance between two vertices is just the electrical resistance in the network between them. It equals the potential difference induced between the vertices when a unit current is injected at one and extracted at the other. In terms of the Laplacian quadratic form, the effective resistance between vertices u and v may be written as

$$R_{(u,v)} = \left(\min_{x: x(u)=1, x(v)=0} Q_G(x)\right)^{-1},$$

an identity which follows from the well-known energy minimizing property of electrical flows.

The Laplacian quadratic form provides a natural way to solve regression problems on graphs. In these problems, one is told the values of x on some subset of the nodes S and is asked to infer the values on the remaining nodes. One approach to solving these problems is to view the known values as voltages that have been fixed, and the values at the other nodes as the induced voltages. That is, one seeks the vector x that minimizes $Q_G(x)$ while agreeing with the known values.[36] One can show that if two graphs are spectrally similar, then the solutions to all such regression problems on the graphs will be similar as well.

The problems of regression and computing effective resistances are special cases of the problem that motivated the definition of spectral similarity: the solution of linear equations in Laplacian matrices. The Laplacian quadratic form can be written as

$$Q_G(x) = x^T L_G x,$$

where L_G is the Laplacian matrix of G. The Laplacian matrix of a graph $G = (V, w)$ is defined by

$$L_G(u, v) \overset{\text{def}}{=} \begin{cases} -w_{(u,v)} & \text{if } u \ne v \\ \sum_z w_{(u,z)} & \text{if } u = v. \end{cases}$$

The problem of solving systems of linear equations in Laplacian matrices arises in many areas of computational science and optimization. In fact, the spectral similarity measure is *identical* to the concept of relative condition number in numerical linear algebra. If two graphs are spectrally similar, then through the technique of preconditioning one can use solutions to linear equations in the Laplacian matrix of one graph to solve systems of linear equations in the Laplacian of the other.

2.3. Spectral similarity of matrices

For two symmetric matrices A and B in $\mathbb{R}^{n \times n}$, we write $A \preccurlyeq B$ to indicate that

$$x^T A x \le x^T B x, \quad \text{for all } x \in \mathbb{R}^n.$$

We say A and B are *σ-spectrally similar* if

$$B/\sigma \preccurlyeq A \preccurlyeq \sigma \cdot B. \tag{2}$$

We have named this relation spectral similarity because it implies that the two matrices have similar eigenvalues. The Courant-Fisher Theorem tells us that

$$\lambda_i(A) = \max_{S:\dim(S)=i} \min_{x \in S} \frac{x^T A x}{x^T x}.$$

Thus, if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A and $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ are the eigenvalues of B, then for all i, $\lambda_i/\sigma \leq \tilde{\lambda}_i \leq \sigma \cdot \lambda_i$.

Using this notation, we can now write inequality (1) as

$$L_{\tilde{G}}/\sigma \preccurlyeq L_G \preccurlyeq \sigma \cdot L_{\tilde{G}}. \tag{3}$$

That is, two graphs are σ-spectrally similar if their Laplacian matrices are. We remark that the Laplacian matrix of a graph is (i) *symmetric*, that is, $L_G = L_G^T$, (ii) *positive semi-definite*, that is, all eigenvalues of L_G are non-negative, and (iii) *(weakly) diagonally dominant*, that is, for all i, $L_G(i,i) \geq \sum_{j \neq i} |L_G(i,j)|$. From consideration of the Laplacian quadratic form, it is easy to verify that if G is connected, then the null space of L_G is just the span of the all 1's vector. Thus all connected graphs have the same Laplacian null space and exactly one zero eigenvalue.

2.4. Distance similarity

It is worth mentioning an interesting alternative to cut- and spectral-similarity. If one assigns a length to every edge in a graph, then these lengths induce a shortest-path distance between every pair of vertices. We say that two different graphs on the same vertex set are *σ-distance similar* if the distance between each pair of vertices in one graph is within a multiplicative factor of σ of the distance between the corresponding pair of vertices in the other graph. Formally, if G and $\tilde{G}$ are the graphs and if $\text{dist}_G(u, v)$ is the distance between vertices u and v in G, then G and $\tilde{G}$ are *σ-distance similar* if for all $u, v \in V$,

$$\text{dist}_G(u,v)/\sigma \leq \text{dist}_{\tilde{G}}(u,v) \leq \sigma \cdot \text{dist}_G(u,v).$$

When $\tilde{G}$ is a subgraph of G, the inequality $\text{dist}_G(u,v) \leq \text{dist}_{\tilde{G}}(u,v)$ is automatically satisfied. Peleg and Ullman[29] defined a t-spanner of a graph G to be a subgraph such that for all $u, v \in V$,

$$\text{dist}_{\tilde{G}}(u,v) \leq t \cdot \text{dist}_G(u,v).$$

They were interested in finding sparse t-spanners. It has been shown[4] that every weighed graph has a $(2t+1)$-spanner with $O(n^{1+1/t})$ edges. The most extreme form of a sparse spanner is the *low stretch spanning tree*, which has only $n - 1$ edges, but which only approximately preserves distances on average,[1] up to polylogarithmic distortion.

3. FINDING SPARSE SUBSTITUTES

A (σ, d)-*spectral sparsifier* of a graph G is a graph $\tilde{G}$ satisfying

1. $\tilde{G}$ is σ-spectrally similar to G
2. The edges of $\tilde{G}$ consist of reweighted edges of G
3. $\tilde{G}$ has at most $d|V|$ edges

Since spectral similarity implies that the total edge weight of a graph is preserved, the spectral sparsifier can only have fewer edges than G if those edges have larger weights.

In this section, we begin by discussing sparsifiers of the complete graph, which have been known for some time. We then describe a sequence of ever-stronger existence theorems, culminating in the statement that any graph G has a $(1 + \epsilon, 4/\epsilon^2)$-spectral sparsifier for every $\epsilon \in (0, 1)$.

3.1. Cliques have constant-degree sparsifers

To warm up, let us first examine the quality of the hypercube as a spectral sparsifier of the complete graph. Assume for convenience that n is a power of two. Let G be the complete graph on n vertices. All the non-zero eigenvalues of L_G equal n, so for every unit vector x orthogonal to the all-1s vector,

$$x^T L_G x = n.$$

The non-zero eigenvalues of the Laplacian of the hypercube with n vertices in which every edge has weight 1 are $(2, \ldots, 2 \log n)$. Let H be this hypercube, but with edge weights $n/(2\sqrt{\log n})$. The non-zero eigenvalues of the Laplacian of H are then

$$\left(n/\sqrt{\log n}, \ldots, n\sqrt{\log n}\right)$$

which implies that for every unit vector x orthogonal to the all-1s vector,

$$x^T L_H x \in \left[n/\sqrt{\log n}, n\sqrt{\log n}\right].$$

Thus, H is a $(\sqrt{\log n}, \log n/2)$-spectral sparsifier of the n-clique.

In fact, the complete graph has much better spectral sparsifiers. Consider the Ramanujan graphs,[26, 27] which are d-regular graphs all of whose non-zero Laplacian eigenvalues lie between $d - 2\sqrt{d-1}$ and $d + 2\sqrt{d-1}$. If we let $\tilde{G}$ be a Ramanujan graph with edge weight n/d, then for every unit vector x orthogonal to the all-1s vector,

$$x^T L_{\tilde{G}} x \in \left[n - 2n\sqrt{d-1}/d,\ n + 2n\sqrt{d-1}/d\right].$$

Thus, $\tilde{G}$ is a $\left((1 - 2\sqrt{d-1}/d)^{-1},\ d/2\right)$-spectral sparsifier of the complete graph G.

3.2. Sampling and decomposition: every graph has a good sparsifier

Ramanujan graphs are members of the family of *expander graphs*. These are sparse graphs in which every subset of vertices has a significant fraction of its edges connecting to vertices outside the subset (see below for details). As with the hypercube and Ramanujan graphs, we can show that every expander can be rescaled to be a good spectral sparsifier of the complete graph.

It is well known that random sparse graphs are usually good expanders (see Bollobás[9] and Friedman[17]). Therefore, one can obtain a good spectral sparsifier of the complete graph by random sampling. Spielman and Teng[34] took this view one step further to show that graph sampling can be used to obtain a good spectral sparsifier for every graph. Their construction was strongly motivated by the work of Benczúr and Karger[8] and Achlioptas and McSherry.[2]

The sampling procedure involves assigning a probability $p_{u,v}$ to each edge $(u, v) \in G$, and then selecting edge (u, v) to be in the graph $\tilde{G}$ with probability $p_{u,v}$. When edge (u, v) is chosen to be in the graph, we multiply its weight by $1/p_{u,v}$.

This procedure guarantees that

$$\mathbf{E}[L_{\tilde{G}}] = L_G.$$

The key step in this approach is to determine the sampling probability $p_{u,v}$ for each edge; there is a tension between choosing small $p_{u,v}$ to generate a sparser $\tilde{G}$ and choosing larger $p_{u,v}$ to more accurately approximate G. Spielman and Teng recognized that some edges are more essential than others, and used a graph decomposition process to implicitly identify these edges and set the sampling probabilities accordingly.

Conductance and graph decomposition. For an unweighted graph $G = (V, E)$ and disjoint subsets $S, T \subset V$, we let $E(S, T)$ denote the set of edges in E connecting one vertex of S with one vertex of T. We define $\mathrm{Vol}(S) = \sum_{i \in S} d_i$ and observe that $\mathrm{Vol}(V) = 2|E|$. We define the *conductance* of a set S of vertices to be

$$\Phi(S) \overset{\text{def}}{=} \frac{|E(S, V - S)|}{\min(\mathrm{Vol}(S), \mathrm{Vol}(V - S))},$$

and we define

$$\Phi_G \overset{\text{def}}{=} \min_{S \subset V} \Phi(S).$$

The *normalized Laplacian* matrix of a graph (see [14]), is defined to be

$$\mathcal{L}_G = D^{-1/2} L_G D^{-1/2},$$

where D is the diagonal matrix whose u-th entry is d_u. The discrete version[3, 31] of Cheeger's[11] inequality (Theorem 2) relates the second eigenvalue of the normalized Laplacian to the conductance of a graph.

Theorem 2 (Discrete Cheeger Inequality).

$$2\Phi_G \geq \lambda_2(\mathcal{L}_G) \geq \Phi_G^2/2.$$

We define a *decomposition* of G to be a partition of V into sets $(A_1, \ldots, A_k)$, for some k. The *boundary* of a decomposition $(A_1, \ldots, A_k)$ is then the set of edges between different vertex sets in the partition:

$$E \cap \cup_{i \neq j} (A_i \times A_j).$$

We say that the decomposition is a *ϕ-decomposition* if $\Phi_{G(A_i)} \geq \phi$ for all i, where $G(A_i)$ denotes the subgraph induced on A_i. It is a *λ-spectral decomposition* if the smallest non-zero normalized Laplacian eigenvalue of $G(A_i)$ is at least λ, for all i. By Cheeger's inequality, every ϕ-decomposition is a $(\phi^2/2)$-spectral decomposition.

The following two theorems (see Spielman and Teng[34]) together imply that for all ϵ, every graph has a $((1 + \epsilon), O(\epsilon^{-2} \log^7 n))$-spectral sparsifier.

Theorem 3 (Spectral Decomposition). *Every G has an $\Omega(\log^{-2} n)$-spectral decomposition with boundary size at most $|E|/2$.*

Theorem 4 (Sampling works for Expanders). *Suppose $\epsilon \in (0, 1/2)$ and $G = (V, E)$ is an unweighted graph with smallest non-zero normalized Laplacian eigenvalue at least λ. Let $\tilde{G} = (V, \tilde{E}, \tilde{w})$ be a graph obtained by sampling the edges of G with probabilities*

$$p_e = \min(1, C/\min(d_u, d_v)) \qquad \text{for each edge } e = (u, v),$$

where

$$C = \Theta((\log n)^2 (\epsilon\lambda)^{-2})$$

and setting weights $\tilde{w}_{(e)} = 1/p_e$ for $e \in \tilde{E}$. Then, with probability at least $1/2$, $\tilde{G}$ is a $(1 + \epsilon)$-spectral approximation of G, and the average degree of $\tilde{G}$ is $O((\log n)^2 (\epsilon\lambda)^{-2})$

To construct a spectral sparsifier of an arbitrary unweighted graph, we first apply Theorem 3 to find a $\Omega(\frac{1}{\log^2 n})$-spectral decomposition of the graph in which the boundary has at most half the edges. We then sparsify each of the components by random sampling, and we sparsify the graph formed by the boundary edges recursively. Adding the sparsifiers obtained yields a sparsifier for the original graph, as desired.

3.3. Sampling by effective resistance

By using effective resistances to define the edge sampling probabilities p_e, Spielman and Srivastava[32] proved that every graph has a $((1 + \epsilon), O(\log n/\epsilon^2))$-spectral sparsifier. These spectral sparsifiers have a similar number of edges to the cut sparsifiers described in Theorem 1, and many fewer edges than those produced by Spielman and Teng[34]. We define R_e, the effective resistance of an edge e, to be the effective resistance between its endpoints. It is well-known that R_e is proportional to the commute time between the end-vertices of e,[10] and is equal to the probability that e appears in a random spanning tree of G. Spielman and Srivastava proved that sampling with edge probability p_e proportional to $w_e R_e$ is the "right" distribution for creating spectral sparsifiers.

Theorem 5 (Sampling by Effective Resistance). *For any weighted graph $G = (V, E, w)$ and $0 < \epsilon \leq 1$, let $\tilde{G}$ be the graph obtained by the following random process:*

> Set $q = 8n \log n/\epsilon^2$. Choose a random edge of G with probability p_e proportional to $w_e R_e$, and add e to the edge set of $\tilde{G}$ with weight w_e/qp_e. Take q samples independently with replacement, summing weights if an edge is chosen more than once.

Then with probability at least $1/2$, $\tilde{G}$ is a $(1 + \epsilon)$-spectral approximation of G.

The proof of Theorem 5 is matrix-analytic. We begin by observing that the Laplacian matrix of G can be expressed as a sum of outer products of vectors:

$$L_G = \sum_{(u,v)\in E} w_{(u,v)}(e_u - e_v)(e_u - e_v)^T,$$

where e_u denotes the elementary unit vector in direction u. Since the edges in $\tilde{G}$ are a subset of the edges in G, its Laplacian may also be expressed as a (differently weighted) sum of the same outer products:

$$L_{\tilde{G}} = \sum_{(u,v)\in E} \tilde{w}_{(u,v)}(e_u - e_v)(e_u - e_v)^T.$$

Suppose the non-zero eigenvalue-and-eigenvector pairs of L_G are $(\lambda_1, u_1), \ldots, (\lambda_{n-1}, u_{n-1})$. Then we can write

$$L_G = \sum_{i=1}^{n-1} \lambda_i u_i u_i^T.$$

Let L_G^+ be the Moore-Penrose Pseudoinverse of L_G, that is,

$$L_G^+ = \sum_{i=1}^{n-1} u_i u_i^T / \lambda_i.$$

Then $L_G L_G^+ = L_G^+ L_G = \sum_i u_i u_i^T \overset{\text{def}}{=} \Pi$ is the projection matrix onto the span of $\{u_i\}$.

The key to the analysis of Theorem 5, and the improved construction of Section 3.4, is the observation that

$$L_G/\sigma \preceq L_{\tilde{G}} \preceq \sigma L_G \Leftrightarrow \prod/\sigma \preceq L_G^{+/2} L_{\tilde{G}} L_G^{+/2} \preceq \sigma \prod,$$

where $L_G^{+/2}$ is the square root of L_G^+. We will show that the sampling procedure described is likely to satisfy this latter condition. To this end, define random variables $\{s_e\}_{e\in E}$ to capture the outcome of the sampling procedure:

$$s_e \overset{\text{def}}{=} \frac{\tilde{w}_e}{w_e} = \frac{(\#\text{ of times } e \text{ is sampled})}{qp_e}. \qquad (4)$$

Then

$$L_{\tilde{G}} = \sum_{(u,v)\in E} s_{(u,v)} w_{(u,v)} (e_u - e_v)(e_u - e_v)^T$$

and $\mathbf{E}[s_{(u,v)}] = 1$ for all $(u, v) \in E$, whence $\mathbf{E}[L_{\tilde{G}}] = L_G$. We now write:

$$L_G^{+/2} L_{\tilde{G}} L_G^{+/2} = \sum_{(u,v)\in E} s_{(u,v)} w_{(u,v)} L_G^{+/2}(e_u - e_v)(e_u - e_v)^T L_G^{+/2}$$
$$= \frac{1}{q}\sum_{i\le q} Y_i Y_i^T,$$

where the Y_i are i.i.d. random vectors sampled from the distribution

$$Y = \frac{1}{\sqrt{p_{(u,v)}}}\left(w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right) \quad \text{with probability } p_{(u,v)}.$$

Notice that $\mathbf{E}[YY^T] = \Pi$ so the expectation of each term is correct. To analyze the sum, we apply the following concentration theorem for sums of independent rank one matrices due to Rudelson.[30]

Theorem 6 (Rudelson). *Let p be a probability distribution over $\Omega \subset \mathbb{R}^d$ such that $\sup_{y\in\Omega} \|y\|_2 \le M$ and $\|\mathbf{E}[yy^T]\| \le 1$. Let $y_1, \ldots, y_q$ be independent samples drawn from p with replacement. Then,*

$$\mathbf{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q} y_i y_i^T - \mathbf{E}\left[yy^T\right]\right\|_2\right] \le \min\left(8M\sqrt{\frac{\log q}{q}}, 1\right).$$

To optimize the application of Rudelson's theorem, we choose the $\{p_{(u,v)}\}$ so that all possible values of Y have the same norm, that is,

$$\|Y\| = \left\|\frac{1}{\sqrt{p_{(u,v)}}}(w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v))\right\| = \gamma$$

for some fixed γ, for all $(u, v) \in E$. An elementary calculation reveals that the value of γ that causes the sum of the probabilities to be one is $\sqrt{n-1}$. Thus if we sample according to probabilities

$$p_{u,v} = \frac{\left\|w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right\|^2}{n-1},$$

then we can take $M = \sqrt{n-1}$ in Theorem 6. This tells us that $q = O(n \log n/\epsilon^2)$ samples are sufficient to obtain a $(1 + \epsilon)$-spectral approximation with high probability, from which Theorem 5 follows.

As stated earlier, the probabilities we have chosen for sampling edges have a natural meaning:

$$\left\|w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right\|^2 = w_{(u,v)}(e_u - e_v) L_G^+(e_u - e_v) = w_{(u,v)} R_{(u,v)},$$

where

$$R_{(u,v)} \overset{\text{def}}{=} (e_u - e_v) L_G^+(e_u - e_v)$$

is the effective resistance between the vertices u and v.

3.4. Twice-Ramanujan sparsifiers

In a nutshell, Spielman and Srivastava first reduced the sparsification of $G = (V, E, w)$ to the following algebraic problem: compute scalars $\{s_{u,v} \ge 0 | (u, v) \in E\}$ such that $\tilde{E} = \{e | s_{u,v} > 0\}$ has small cardinality, and

$$(1-\epsilon)\prod \preccurlyeq L_G^{+/2} L_{\tilde{G}} L_G^{+/2} \le (1+\epsilon)\prod.$$

They then applied sampling, based on effective resistances, to generate the $\{s_{u,v}\}$ and $\tilde{E}$.

Batson et al.[7] gave a deterministic polynomial-time algorithm for computing $\{s_e\}$ and $\tilde{E}$, and obtained the following theorem, which is essentially the best possible result for spectral sparsification.

Theorem 7 (Batson-Spielman-Srivastava). *For every $d > 1$, every undirected, weighted n-node graph $G = (V, E, w)$ has a*

$$\left(\frac{\sqrt{d}+1}{\sqrt{d}-1}, d\right) \text{- spectral sparsifier}.$$

In particular, G has a $((1 + 2\epsilon), 4/\epsilon^2)$-spectral sparsifier, for every $0 < \epsilon < 1$.

At the heart of their construction is the following purely linear algebraic theorem, which may be shown to imply Theorem 7 by an argument similar to that in Section 3.3.

Theorem 8. *Suppose $d > 1$ and $v_1, \ldots, v_m \in \mathbb{R}^n$ satisfy $\sum_{i=1}^{m} v_i v_i^T = I_n$, where I_n is the $n \times n$ identity matrix. Then, there exist scalars $s_i \ge 0$ with $|\{i : s_i \ne 0\}| \le dn$ such that*

$$I_n \preccurlyeq \sum_{i=1}^{m} s_i v_i v_i^T \preccurlyeq \left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)^2 \cdot I_n$$

The proof for Theorem 8 builds the sum $\sum_i s_i v_i v_i^T$ iteratively, by adding one vector at a time. For $\tilde{m} = dn$, it chooses a sequence of vectors $\pi(1), \ldots, \pi(\tilde{m})$ and weights $s_{\pi(1)}, \ldots, s_{\pi(\tilde{m})}$, which in turn defines a sequence of matrices $0 = A_0, \ldots, A_{\tilde{m}}$, where $A_t = \sum_{i=1}^{t} s_{\pi(i)} v_{\pi(i)} v_{\pi(i)}^T$. Observe that $A_t = A_{t-1} + s_{\pi(t)} v_{\pi(t)} v_{\pi(t)}^T$ and we always have $A_t \succeq A_{t-1}$. The goal is to control the eigenvalues of A_t at each step and guarantee that they grow with t in a *steady* manner, so that the final matrix $A_{\tilde{m}} = A_{dn}$ has all eigenvalues within a constant factor $\left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)^2$ of each other and is hence a good approximation to the identity.

Batson, Spielman, and Srivastava use two "barrier" potential functions to guide their choice of $\pi(i)$ and $s_{\pi(i)}$ and ensure steady progress. Specifically, for $u, l \in \mathbb{R}$, and A a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$, they define

$$\Phi^u(A) \overset{\text{def}}{=} \mathrm{Tr}(uI - A)^{-1} = \sum_i \frac{1}{u - \lambda_1} \quad \text{(Upper potential)}$$

$$\Phi_l(A) \overset{\text{def}}{=} \mathrm{Tr}(A - uI)^{-1} = \sum_i \frac{1}{\lambda_1 - l} \quad \text{(Lower potential)}$$

When $l \cdot I_n \prec A \prec u \cdot I_n$, small values of these potentials indicate that the eigenvalues of A do not cluster near u or l. This turns out to be a sufficient induction hypothesis to sustain the following iterative process:

(1) Begin by setting the lower barrier l, to $-n$ and the upper barrier, u to n. It can be checked that both potentials are bounded by 1. (2) At each step, increase the upper barrier u by a fixed constant δ_u and the lower barrier l by another fixed constant $\delta_l < \delta_u$. It can then be shown that as long as the potentials remain bounded, there must exist at every time t a choice of a vector $v_{\pi(i)}$ and a weight $s_{\pi(i)}$ so that the addition of $s_{\pi(i)} v_{\pi(i)} v_{\pi(i)}^T$ to A_{t-1} and the increments $l \rightarrow l + \delta_l$ and $u \rightarrow u + \delta_u$ *do not increase either potential* and keep all the eigenvalues $\lambda_i(A_t)$ between the barriers. Iterating the above process ensures steady growth of all the eigenvalues and yields Theorem 8.

3.5. Some extensions

In a recent work, de Carli Silva et al.[15] extended spectral sparsification to the sums of positive semidefinite matrices that have arbitrary rank. They proved the following theorem.

Theorem 9. *Let $B_1, \ldots, B_m$ be symmetric (or Hermitian) positive semidefinite $n \times n$ matrices and $B = \sum_{i=1}^{m} B_i$. Then for any $\epsilon \in (0, 1)$, there exist nonnegative $s_1, \ldots, s_m$, at most $O(n/\epsilon^2)$ of which are nonzero, such that*

$$B \preccurlyeq \sum_i s_i B_i \preccurlyeq (1+\epsilon) \cdot B.$$

Moreover, $\{s_1, \ldots, s_m\}$ can be computed in $O(mn^3/\epsilon^2)$ time.

Another extension is subgraph spectral sparsification,[22] in which one is given a union of two graphs G and W and an integer κ, and asked to find a κ-edge graph W_κ such that $G + W_\kappa$ is a good spectral sparsifier of $G + W$. When combined with the best-known construction of low-stretch spanning trees,[1] this provides nearly optimal *ultra-sparsifiers*.

Theorem 10 (Kolla, Makarychev, Saberi, Teng). *For each positive integer κ, every n-vertex graph has an $O(\sqrt{(n/\kappa)}\, \log n\, \log n\, \log n)$-spectral approximation with at most $(n - 1 + \kappa)$-edges.*

Ultra-sparsifiers have so few edges that they have a large number of vertices of degree 1 or 2. They are a key component of the algorithms for solving linear equations described in Section 4.1.

4. ALGORITHMIC APPLICATIONS

4.1. Numerical algorithms: Laplacian systems

One of the most fundamental computational problems is that of solving a system of linear equations. One is given an $n \times n$ matrix A and an n-dimensional vector b, and is asked to find a vector x such that $Ax = b$. It is well known that every linear system can be solved by the classic Gaussian elimination method in polynomial time. However, Gaussian elimination usually takes super-linear or even super-quadratic time in the number of non-zero entries of A, making its use impractical for large matrices.

In many real-world applications, the matrix A is sparse and one only requires an approximate solution to the linear system. For example, given a precision parameter ϵ, we may be asked to produce an $\tilde{x}$ such that $\|A\tilde{x} - b\|_2 \leq \epsilon \|b\|_2$. For sparse positive semi-definite linear systems the fastest general purpose algorithm is the Conjugate Gradient (CG). It essentially solves $Ax = b$ by multiplying a sequence of vectors by A. As the multiplication of a vector by A takes time proportional to the number of non-zero entries in A, CG can run quickly when A is sparse The number of matrix-vector products performed by CG depends on the *condition number* $\kappa(A)$ of A, the ratio of its largest eigenvalue to its smallest eigenvalue. It is well-known in numerical analysis[19] that it is sufficient to compute $O(\sqrt{\kappa(A)} \log(1/\epsilon))$ matrix-vector products to find a solution of accuracy ϵ.

Preconditioning is the strategy of finding a relatively easily invertible matrix B which is σ–spectrally similar to A, and solving the related system $B^{-1}Ax = B^{-1}b$. In each iteration, the preconditioned CG algorithm solves a linear system in B and performs a matrix-vector product in A, and only $O(\sqrt{\kappa(B^{-1}A)}\, \log(\epsilon^{-1})) = O(\sigma \log(\epsilon^{-1}))$ iterations are required. If it is easy to solve systems of linear equations in B, then the cost of each iteration is small and this algorithm will run quickly.

In 1990, Vaidya brought graph theory into the picture. Using preconditioners consisting of a maximum spanning tree of a graph plus a small number of carefully chosen edges, Vaidya obtained an $O(m^{1.75} \log(1/\epsilon))$-time algorithm for solving linear systems in Laplacian matrices with m non-zero entries. The exponent was still too large to be practical, but the idea was powerful. Spielman and Teng[33] were able to enhance Vaidya's approach with spectral sparsifiers and low-stretch spanning trees to obtain the first nearly linear time algorithm for solving Laplacian linear systems.

Theorem 11 (Spielman-Teng). *Linear systems in a graph Laplacian L_G can be solved to precision ϵ in time $O(m \log^{O(1)} n \log(1/\epsilon))$.*

In spite of its strong asymptotic behavior, the large exponent on the log factor makes this algorithm slow in practice. The Spielman-Srivastava sparsification algorithm offered no improvement—effective resistances do give the ideal probabilities with which to sample edges for a sparsifier, but computing them requires solving yet more Laplacian linear systems.

Koutis et al.[24] removed this dependency problem by using low-stretch trees to compute less aggressive sampling probabilities which are strictly greater than those suggested by effective resistances. This can be done more quickly, and along with some other elegant ideas and fast data structures, is sufficient to yield a Laplacian linear system solver which runs in time

$$O(m \log n \log \log^2 n \log(1/\epsilon)).$$

4.2. Fast sparsification algorithms

The algorithms from Section 3 certified the existence of good sparsifiers, but run quite slowly in practice. Those techniques have been significantly refined, and now there are three major ways to produce sparsifiers quickly.

First, the bottleneck in sampling via effective resistances is approximating the effective resistances themselves. The Laplacian system solver of Koutis, Miller, and Peng described above can be used to calculate those resistances, which can then be used to sample the graph. The best analysis is given by Koutis et al.[23] who give an $O(m \log n \log \log n \log(1/\epsilon))$ time algorithm for generating $((1+\epsilon), O(\log^3 n/\epsilon^2))$-spectral sparsifiers.

Second, the decomposition-and-sampling algorithm of Spielman and Teng[34] can be sped up by improving the local clusterings used to create a decomposition. In the local clustering problem, one is given a vertex and cluster size as input, and one tries to find a cluster of low conductance near that vertex of size at most the target size, in time proportional to the target cluster size. Faster algorithms for local clustering have been developed by Andersen et al.[5] and by Andersen and Peres.[6]

Third, unions of random spanning trees of a graph G can make good cut-sparsifiers: the union of two random spanning trees is $(\log n)$-cut similar to G,[20] while the union of $O(\log^2 n/\epsilon^2)$ random spanning trees, reweighed proportionally to effective resistance, is $(1+\epsilon)$-cut similar to G.[18] Although it remains to be seen if the union of a small number of random spanning trees can produce a spectral sparsifier, Kapralov and Panigrahy showed that one can build a $(1+\epsilon)$-spectral sparsifier of a graph from the union of spanners of $O(\log^4 n/\epsilon^4)$ random subgraphs of G.[21]

4.3. Network algorithms

Fast algorithms for spectral sparsification and Laplacian systems provide a set of powerful tools for network analysis. In particular, they lead to nearly-linear time algorithms for the following basic graph problems:

APPROXIMATE FIEDLER VECTORS[33]: *Input*: $G = (V, E, w)$ and $\epsilon > 0$. *Output*: an ϵ-approximation of the second smallest eigenvalue, $\lambda_2(L_G)$, (also known as the Fiedler value) of L_G, along with a vector v orthogonal to the all 1s vector such that $v^T L_G v \leq (1+\epsilon)\lambda_2(L_G)$.

ELECTRICAL FLOWS[13, 32, 33]: *Input*: $G = (V, E, w)$ where weights are resistances, $s, t \in V$, and $\epsilon > 0$. *Output*: an ϵ-approximation of the electrical flows over all edges when 1 unit of flow is injected into s and extracted from t.

EFFECTIVE RESISTANCE APPROXIMATION[32]: *Input*: $G = (V, E, w)$ where weights are resistances and $\epsilon > 0$. *Output*: a data structure for computing an ϵ-approximation of the effective resistance of any pair of vertices in G. Data Structure: $O(n \log n/\epsilon^2)$ space, and $O(\log/\epsilon^2 n)$ query time, and $O(m \log^2 n \log \log^2 n/\epsilon^2)$ preprocessing time.

LEARNING FROM LABELED DATA ON A GRAPH[35]: *Input* a strongly connected (aperiodic) directed graph $G = (V, E)$ and a labeling function y, where y assigns a label from a label set $Y = \{1, -1\}$ to each vertex of a subset $S \subset V$ and 0 to vertices in $V - S$, and a parameter μ. *Output* the function $f: V \rightarrow \mathbb{R}$ that minimizes

$$\Omega(f) + \mu \|f - y\|^2,$$

where

$$\Omega(f) = \sum_{(u,v)\in E} \pi(u) p(u,v) (f(u)\pi(u)^{-1/2} - f(v)\pi(v)^{-1/2}),$$

$$p(u,v) = \begin{cases} 1/(\text{out} - \text{degree of } u) & \text{for } (u,v) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

and π is the stationary distribution of the random walk on the graph with transition probability function p.

COVER TIME OF RANDOM WALK[16]: *Input*: $G = (V,E)$, *Output*: a constant approximation of the cover time for random walks.

The algorithm for ELECTRICAL FLOWS led to a breakthrough for the following fundamental combinatorial optimization problem, for which the best previously known algorithm ran in time $\tilde{O}(mn^{1/2}/\epsilon)$, $\tilde{O}(mn^{2/3} \log \epsilon^{-1})$ and $\tilde{O}(m^{3/2} \log \epsilon^{-1})$.

MAXIMUM FLOWS and MINIMUM CUTS[13]: *Input:* $G = (V, E, w)$ where w are capacities, $s, t \in V$, and $\epsilon > 0$, *Output*: an ϵ-approximation of s-t maximum flow and minimum cut. *Algorithm*: $\tilde{O}(mn^{1/3} \cdot \text{poly}(1/\epsilon))$ time.

Spectral graph sparsification also played a role in understanding other network phenomena. For example, Chierichetti et al.[12] discovered a connection between rumor spreading in a network and the spectral sparsification procedure of Spielman and Teng,[34] and applied this connection to bound the speed of rumor spreading that arises in social networks.

5. OPEN QUESTION

The most important open question about spectral sparsification is whether one can design a nearly linear time algorithm that computes (σ, d)-spectral sparsifiers for any constants σ and d. The algorithms based on Theorem 7 are polynomial time, but slow. All of the nearly-linear time algorithms of which we are aware produce sparsifiers with d logarithmic in n.

6. CONCLUSION

Spectral sparsification has proved to be a remarkably useful tool in algorithm design, linear algebra, combinatorial optimization, machine learning, and network analysis. Theorem 8 has already been applied many times within pure mathematics (see, e.g., Naor[28]). We hope this body of work will encourage more exchange of ideas between numerical analysis, pure mathematics and theoretical computer science, and inspire and enable the development of faster algorithms and novel analyses of network phenomena.

References

1. Abraham, I., Neiman, O. Using petal-decompositions to build a low stretch spanning tree. In (2012).
2. Achlioptas, D., Mcsherry, F. Fast computation of low-rank matrix approximations. , 2 (2007), 9.
3. Alon, N., Milman, V.D. λ_1, isoperimetric inequalities for graphs, and superconcentrators. , 1 (1985), 73–88.
4. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J. On sparse spanners of weighted graphs. (1993), 81–100, 10.1007/BF02189308.
5. Andersen, R., Chung, F., Lang, K. Local graph partitioning using pagerank vectors. In (2006), 475–486.
6. Andersen, R., Yuval, P. Finding sparse cuts locally using evolving sets. In (2009), ACM, 235–244.
7. Batson, J.D., Spielman, D.A., Srivastava, N. Twice-Ramanujan sparsifiers. 6 (2012), 1704 –1721.
8. Benczúr, A.A., Karger, D.R. Approximating s-t minimum cuts in O(n^2) time. In (1996), 47–55.
9. Bollobás, B. The isoperimetric number of random regular graphs. , 3 (May 1988), 241–244.
10. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R., Tiwari, P. The electrical resistance of a graph captures its commute and cover times. In (1989), ACM, New York, NY, USA, 574–586.
11. Cheeger, J. A lower bound for smallest eigenvalue of Laplacian. In (1970), Princeton University Press, 195–199.
12. Chierichetti, F., Lattanzi, S., Panconesi, A. Rumour spreading and graph conductance. In (2010), 1657–1663.
13. Christiano, P., Kelner, J.A., Madry, A., Spielman, D.A., Teng, S.H. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In (2011), 273–282.
14. Chung, F.R.K. Spectral graph theory. CBMS Regional Conference Series in Mathematics, American Mathematical Society, 1997.
15. de Carli Silva, M.K., Harvey, N.J.A., Sato, C.M. Sparse sums of positive semidefinite matrices. (2011), abs/1107.0088.
16. Ding, J., Lee, J.R., Peres, Y. Cover times, blanket times, and majorizing measures. In (2011), 61–70.
17. Friedman, J. A Proof of Alon's Second Eigenvalue Conjecture and Related Problems. Number 910 in Memoirs of the American Mathematical Society. American Mathematical Society, 2008.
18. Fung, W.S., Hariharan, R., Harvey, N.J.A., Panigrahi, D. A general framework for graph sparsification. In (2011), 71–80.
19. Golub, G.H., Van Loan, C.F. Matrix Computations, 2nd edn, Johns Hopkins University Press, 1989.
20. Goyal, N., Rademacher, L., Vempala, S. Expanders via random spanning trees. In (2009), 576–585.
21. Kapralov, M., Panigrahy, R. Spectral sparsification via random spav nners. In (2012), 393–398.
22. Kolla, A., Makarychev, Y., Saberi, A., Teng, S.H. Subgraph sparsification and nearly optimal ultrasparsifiers. In (2010), 57–66.
23. Koutis, I., Levin, A., Peng, R. Improved spectral sparsification and numerical algorithms for SDD matrices. In (2012), 266–277.
24. Koutis, I., Miller, G.L., Peng, R. A nearly-m log n time solver for SDD linear systems. In (2011), 590–598.
25. Leiserson, C. Fat-trees: universal networks for hardware-efficient supercomputing. , 10 (October1985), 892–901.
26. Lubotzky, A., Phillips, R., Sarnak, P. Ramanujan graphs. , 3 (1988), 261–277.
27. Margulis, G.A. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. , 1 (July 1988), 39–46.
28. Naor, A. Sparse quadratic forms and their geometric applications. (2011), 1033.
29. Peleg, D., Ullman, J.D. An optimal synchronizer for the hypercube. , 4 (1989), 740–747.
30. Rudelson, M. Random vectors in the isotropic position. , 1 (1999), 60–72.
31. Sinclair, A., Jerrum, M. Approximate counting, uniform generation and rapidly mixing Markov chains. 1 (July 1989), 93–133.
32. Spielman, D.A., Srivastava, N. Graph sparsification by effective resistances. 6 (2011), 1913–1926.
33. Spielman, D.A., Teng, S.H. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. (2008), abs/cs/0607105.
34. Spielman, D.A., Teng, S.H. Spectral sparsification of graphs. , 4 (2011), 981–1025.
35. Zhou, D., Huang, J., Scholkopf, B. Learning from labeled and unlabeled data on a directed graph. In (2005), 1041–1048.
36. Zhu, X., Ghahramani, Z., Lafferty, J.D. Semi-supervised learning using Gaussian fields and harmonic functions. In (2003).

Joshua Batson, Mathematics, MIT.

Daniel A. Spielman, Computer Science & Applied Mathematics, Yale University.

Nikhil Srivastava, Microsoft Research, Bangalore.

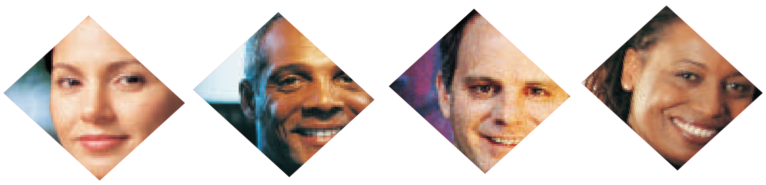
Shang-Hua Teng, Computer Science, USC.

Association for Computing Machinery

Advancing Computing as a Science & Profession

MentorNet

You've come a long way.
Share what you've learned.



ACM has partnered with MentorNet, the award-winning nonprofit e-mentoring network in engineering, science and mathematics. MentorNet's award-winning **One-on-One Mentoring Programs** pair ACM student members with mentors from industry, government, higher education, and other sectors.

- Communicate by email about career goals, course work, and many other topics.
- Spend just **20 minutes a week** - and make a huge difference in a student's life.
- Take part in a lively online community of professionals and students all over the world.

Make a difference to a student in your field.
Sign up today at: www.mentornet.net
Find out more at: www.acm.org/mentornet

MentorNet's sponsors include 3M Foundation, ACM, Alcoa Foundation, Agilent Technologies, Amylin Pharmaceuticals, Bechtel Group Foundation, Cisco Systems, Hewlett-Packard Company, IBM Corporation, Intel Foundation, Lockheed Martin Space Systems, National Science Foundation, Naval Research Laboratory, NVIDIA, Sandia National Laboratories, Schlumberger, S.D. Bechtel, Jr. Foundation, Texas Instruments, and The Henry Luce Foundation.

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Head of the Department of Computer Science and Engineering

The Hong Kong University of Science and Technology (HKUST), opened in October 1991, comprises four Schools: Science, Engineering, Business & Management, and Humanities & Social Science. The University's mission is to advance learning and scholarship; to promote research, development, and entrepreneurship, and to contribute to the region's economic and social development.

The School of Engineering, the largest School of the University, currently enrolls about 38% of the University's total undergraduate and postgraduate students of approximately 12,600. It comprises six departments: Chemical & Biomolecular Engineering, Civil & Environmental Engineering, Computer Science & Engineering, Electronic & Computer Engineering, Industrial Engineering & Logistics Management, and Mechanical Engineering.

The Department of Computer Science & Engineering (CSE) currently has 44 faculty members, teaching about 560 undergraduate students and 180 postgraduate research students. The Department conducts comprehensive teaching and research programs in both basic and applied aspects of Computer Science & Engineering. The academic degrees offered by the Department are: BEng, MSc, MPhil and PhD. Research activities in the Department are broadly categorized into artificial intelligence; data, knowledge and information management; networking and computer systems; software technologies; theoretical computer science; vision and graphics. For more information, please visit the University and Department websites available on http://www.ust.hk/ and http://www.cse.ust.hk/ respectively.

Applications/nominations are invited from well-qualified and accomplished scholars for the position. In addition to extensive teaching and research experience, the successful candidate must have demonstrated leadership qualities necessary to lead and manage the Department in its diverse academic and administrative functions and to interact effectively with industry and commerce.

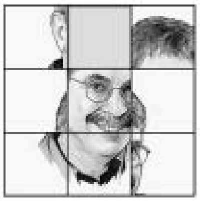
Salary will be highly competitive with generous benefits. Applications/nominations together with detailed curriculum vitae and the names and addresses/fax numbers/email addresses of three referees should be sent to Professor Chung Yee Lee, Chair of Search Committee for Headship of CSE, c/o School of Engineering, HKUST, Clearwater Bay, Kowloon, Hong Kong [Fax No.: (852) 2358 1458, e-mail: dhcse@ust.hk] before **Tuesday, 1 October 2013.**

HKUST is committed to increasing the diversity of its faculty and has a range of family-friendly policies in place.

(Information provided by applicants will be used for recruitment and other employment-related purposes.)

SOUTH DAKOTA STATE UNIVERSITY
Department of Electrical Engineering and Computer Science
Assistant Professor of Computer Science

This is a 9-month renewable tenure track position; open January 1, 2014. An earned Ph.D. in Computer Science or a closely related field is required. Successful candidates must have a strong commitment to academic and research excellence. Candidate should possess excellent and effective written, oral, and interpersonal skills. Primary responsibilities will be to teach in the various areas of computer science, to participate widely in the CS curriculum, and to conduct research in the areas of big data systems, applied mathematical and statistical models to CS applications, and related areas. To apply, visit https://YourFuture.sdbor.edu, search for the position, and follow the electronic application process. For questions on the electronic employment process, contact SDSU Human Resources at (605) 688-4128. For questions on the position, contact Dr. Sung Shin, Search Chair, at (605) 688-6235. SDSU is an AA/EEO employer.



DOI:10.1145/2492007.2492030

Peter Winkler

Puzzled
Wins in a Row

Each of these puzzles involves game-playing strategy. If you are sufficiently clever—and sufficiently unmotivated to work hard at being clever—you can solve them all without resorting to algebra. Here is the premise: You have applied to join a chess club and been told that to qualify you must play three games against Ioana (the last new member), winning two games in a row. "Who gets the white pieces?" you ask and are told you and Ioana alternate and you get to decide whether to start with white or with black.

1. Knowing that the probability of beating Ioana is better with the white pieces (first-move advantage), should you choose white or black for the first game?

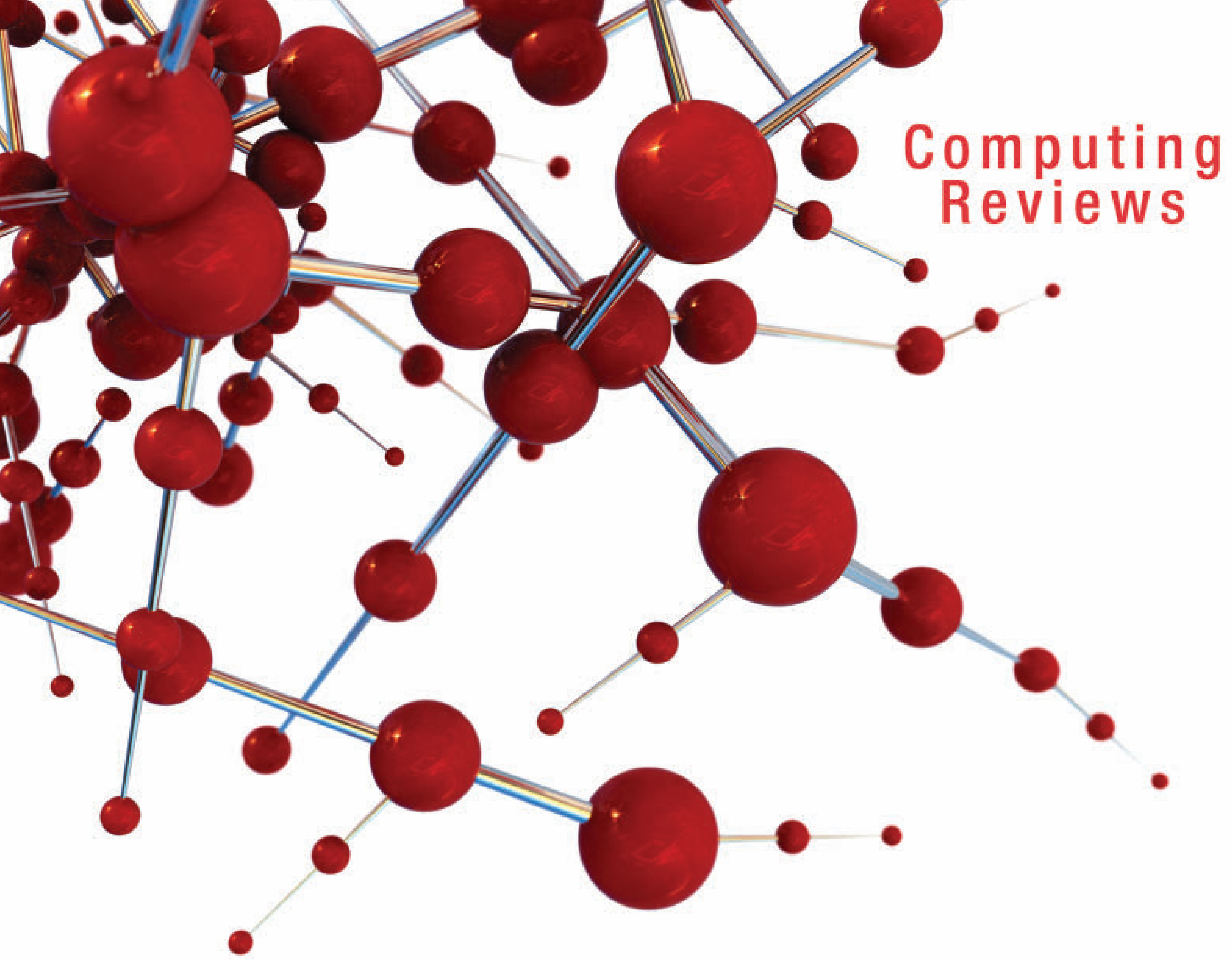
2. Suppose now that prior to the match, you discover Ioana is a former City Champion of Bucharest. Complaining, you persuade the club to give you the following concession: You must still beat Ioana two games in a row, but now you get to play her as many as 17 times, alternating sides as before. Should you choose white or black for the first game?

3. You managed to win two games in a row from Ioana and are now a member of the club. However, to become an officer of the club, you must beat Ioana 10 times in a row, with 49 games to do it. Yipes! This may be more than you can handle, but to maximize your chances, should you start with white or with black?

PHOTOGRAPH BY PAOLO GALLO/PERSEOMEDUSA

Readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science at Dartmouth College, Hanover, NH.



Computing Reviews

CONNECT WITH OUR COMMUNITY OF EXPERTS.

www.computingreviews.com

Association for Computing Machinery

ThinkLoud

They'll help you find the best new books and articles in computing.

Computing Reviews is a collaboration between the ACM and ThinkLoud.

4th Annual ACM SIGPLAN Conference on

Systems, Programming, Languages, Applications: Software for Humanity

Early Registration Deadline
September 27

Location
Hyatt Regency Indianapolis

Events
- 28th Annual OOPSLA
- Onward!
- Wavefront
- Dynamic Languages Symposium (DLS)
- Generative Programming & Component Engineering (GPCE)
- Software Language Engineering (SLE)
- Panels
- Workshops
- Tutorials & Tech Talks
- ...and more

General Chairs
Patrick Eugster & Antony Hosking
Purdue University

OOPSLA Papers Chair
Cristina Lopes
University of California, Irvine

Onward! Papers Chair
Robert Hirschfeld
Hasso-Plattner-Institut Potsdam

Onward! Essays Chair
Bernd Brügge
Technische Universität München

DLS Papers Chair
Carl Friedrich Bolz
Heinrich-Heine-Universität Düsseldorf

More Information
http://splashcon.org
info@splashcon.org

acm Association for Computing Machinery

SPLASH
INDIANAPOLIS 2013
OCTOBER 26-31