ICS NEWS
INFORMS Computing Society Newsletter

Inside This Issue

"One day I'm going to get help for my procrastination problem ..."

ICS 2011
1/9/11–1/11/11

The 12th INFORMS Computing Society Conference will take place from January 9th to January 11th, 2011 at the Monterey Marriott in Monterey, California, USA. Some important dates for the conference are

Author Registration	November 1, 2010
Early Registration	November 1, 2010
Hotel Registration (conference rate)	December 20, 2010

This conference focuses on the interface of computer science, artificial intelligence, operations research, and management science. The conference organizers invite you to submit theoretical and applied work that highlights the conference theme: Operations Research, Computing and Homeland Defense.

General Co-Chairs for the conference are Rob Dell and Kevin Wood, from the Naval Postgraduate School. The conference webmaster is Bill Hart.

Message from the Chair

Robert Vanderbei
Operations Research and Financial Engineering
rvdb@prinecton.edu

As Chair of ICS, one of my duties is to write a brief note for the ICS newsletter on the overall wellbeing of the society. Faced with this task, my first thought was to read what Robin Lougee wrote in a similar context a few years ago. I read the first sentence of her report and thought, hey, that's exactly what I want to say. But, of course, I didn't want to be guilty of plaigerism. Then, I read on and noticed that she was quoting from Dick Barr's report from more than 10 years earlier. So, here, again, is what Dick said more than a decade ago: "The Computer Science Technical Section is in excellent shape, with a healthy treasury, a sizable and active membership, a strong presence at the national INFORMS meetings, its own successful conference series, an excellent journal, an informative newsletter, an established and recognized prize for excellence, and attendees that know how to have a good time." With the exception of a bit of discontinuity with the newsletter and a growth from one prize to three, these words still hold forth today.

(continued on p. 12)

Message from the Editor

Jeff Linderoth
University of Wisconsin-Madison
linderoth@wisc.edu

> "Like every man of sense and good feeling, I abominate work."
> —Aldous Huxley

As the observant member undoubtedly surmised, the society had some trouble in filling the giant shoes of the last ICS Newsletter Editor, Harvey Greenberg. I would like to thank all the authors of this issue for providing me with material, and for their Job-esque patience for this issue to be produced.

> "I have offended God and mankind because my work didn't reach the quality it should have."
> —Leonardo da Vinci

The authors however, have offended neither God nor mankind. In this issue, we have our regular collection of updates—a COIN-OR update from Matt Saltzman, a Mathematical Programming (Optimization?) Glossary update from Allen Holder, and an IJOC update from John Chinneck. We also have a review of Harvey Greenberg's "Myths and Counterexamples" by Jim Orlin and a book review by Todd Munson. Our feature article on GPU Computing is from Sangkyun Lee. I hope the readership finds these features and articles as interesting as I did.

> "If at first you don't succeed, try, try again. Then quit. There's no point in being a damn fool about it."
> —W. C. Fields

This year will mark the end of my less-than-esteemed run as ICS Newsletter Editor. If you are interested becoming the next Newsletter editor, and clearing the very low bar that I have set, please contact me or Bob Vanderbei. I am happy to help the next Newsletter with the transition. And I promise...

> "I'm not going to criticize my successor."
> —George W. Bush

My one piece of advice for the next editor is to visit www.brainyquote.com, as

> "The ability to quote is a serviceable substitute for wit."
> —W. Somerset Maugham

COIN-OR

Matt Saltzman, Clemson University
mjs@clemson.edu

The Computational Infrastructure for Operations Research (COIN-OR, http://www.coin-or.org) is the premier website devoted to open-source software for the operations research community.

The year 2009 was an active one for COIN-OR on many fronts. These are just a few of the achievements of the COIN-OR community.

New Developments in 2009

New projects that went live in 2009 include LEMON, a library of C++ classes for graph and network optimization; ADOL-C, an automatic differentiation library for C and C++; and METSlib, a library for developing metaheuristics.

Optimization Services (OS) now supports the Couenne MINLP solver and offers APIs for solver options and results. OS is now available in a binary distribution for Windows.

A Google Summer of Code project integrated Ipopt with the ASCEND modeling environment to produce an integrated open-source environment for modeling and solving NLPs.

The COIN-OR Strategic Leadership Board has made significant progress in revising the legal policies and procedures for projects. The Technical Leadership Council is developing new infrastructure and build procedures for the projects that use our BuildTools support.

The 2009 COIN Cup was awarded in San Diego to Yuri Levin, Tatsiana Levina, Jeff McGill, Mikhail Nediak and Huseyin Topaloglu, who applied COIN-OR technologies DFO and IPOPT to develop novel techniques for cargo capacity management and dynamic pricing.

The biggest recent COIN news is the retirement of John Forrest from IBM Research and from stewardship of the CLP (COIN-OR Linear Programming) and CBC (COIN-OR Branch and Cut) projects. The new CLP project manager is Julian Hall of University of Edinburgh, and the support team includes Matt Saltzman of Clemson University and Lou Hafer of Simon Fraser University. The CBC project manager is Ted and the support team includes Matt, Lou, Bjarni Kristjannson of Maximal, Dan Fylstra and Edwin Straver of Frontline, and Bill Hart and Cindy Phillips from Sandia.

Plans are under way for events honoring John and celebrating COIN-OR's 10th anniversary at the Austin INFORMS meeting. See https://projects.coin-or.org/Events/wiki/JJHFCOIN10.

Portions of this article appeared in the October 2009 issue of OR/MS Today. Reprinted with permission.

Mathematical Programming Glossary

Allen Holder, Rose-Hulman Institute of Technology
holder@rose-hulman.edu

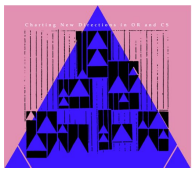
The Mathematics Programming Glossary continues to be an active resource for our community, receiving about 91,000 hits per week. The Glossary is also increasingly cited by other references such as Wikipedia. Over the last several months we have continued with the term-by-term edits needed for the new design; with only six letters remaining. Although not complete, the new version is posted at glossary.computing.society.informs.org/ver2/mpgwiki/. We should have a complete release early in the summer.

The Glossary, under the thankful guidance of Chris Beck, has embarked on a substantial addition of terms relating to constraint programming. This important area had largely been neglected, and over the next year we hope to advance our exposure in this area. A new supplement on complexity theory is also expected over the next few months.

The Glossary houses Harvey Greenberg's "Myths and Counter Examples in Mathematical Programming." The latest revision was posted on February 20, 2010, and it has 213 entries spanning linear programming, integer programming, dynamic programming, nonlinear programming and problems of other special forms. It is a remarkable collection and is being downloaded about 1,100 times per week.

The Computing Society's *Mathematical Programming Glossary* will have an entry here.

Please visit http://glossary.computing.society.informs.org/ to use the *MP Glossary* and learn how you can contribute.


joc.pubs.informs.org/

Message from the Editor of INFORMS Journal on Computing

John Chinneck
Carleton University
joc@mail.informs.org

Things are busy at the INFORMS Journal on Computing! We published 46 papers in the four 2009 issues, covering a wide variety of topics at the interface of operations research and computer science. Highlights included the seven-paper *Special Cluster on High-Throughput Optimization* that appeared in the summer issue, and a survey article on *Evolutionary Algorithms for Vehicle Routing* by Jean-Yves Potvin in the fall issue. The current winter 2010 edition includes a feature article, complete with rejoinders, on *Merging AI and OR to Solve High-Dimensional Stochastic Optimization Problems Using Approximate Dynamic Programming* by Warren Powell, the leading researcher on this topic.

Here are some recent statistics. For the one-year period February 1, 2009 through January 31, 2010 decisions were

rendered on 253 papers. The rate of submission continues to be quite high. There are currently around 90 papers in process.

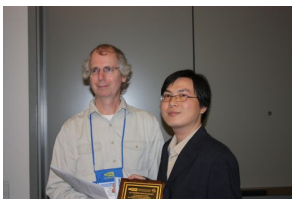
We are fortunate to have support from seven prominent sponsors for 2010: ARKI Consulting and Development, GAMS Development Corporation, Haverly Systems Inc., IBM T.J. Watson Research Center, the INFORMS Computing Society (of course), LINDO Systems Inc., and Palisade Corporation.

2009 also saw changes in personnel, with the addition of a number of knowledgeable new Associate Editors: Sanjeeb Dash of the IBM T.J. Watson Research Center, Antonio Frangioni of the Universitáă di Pisa, Balaji Padmanabhan of the University of South Florida, and David Parkes of the Harvard University, William J. Stewart of North Carolina State University, Daniel Zeng of the University of Arizona. There were also changes in Area Editors as two of our long-time area experts stepped down from their posts. Allen Holder replaced founding JOC Editor-in-Chief Harvey Greenberg as the Area Editor for *Computational Biology and Medical Applications* (recently renamed Applications in Biology, Medicine and Health Care), while Michela Milano of the Universitá di Bologna replaced John Hooker as Area Editor for *Constraint Programming and Optimization*.

As always, we are on the lookout for excellent research at that interesting intersection between operations research and computer science: send us your best work! As a reminder, the journal has nine major areas (Applications in Biology, Medicine and Health Care; Computational Probability and Analysis; Constraint Programming and Optimization; Design and Analysis of Algorithms; Heuristic Search and Learning; Knowledge and Data Management; Modeling: Methods and Analysis; Simulation; and Telecommunications and Electronic Commerce) and well as Feature Articles. You can the Journal online at http://joc.pubs.informs.org.

Wächter and Biegler Win the ICS Prize

Andreas Wächter, of IBM TJ Watson Research Center, and Lorez Beigler, of Carnegie Mellon University were the winners of the 2009 INFORMS Computing Society Prize. for the paper "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming." This paper is the basis of the highly successful code IPOPT for nonlinear programming. This open-source software package, which is one of the most advanced codes for large-scale nonlinear programming,âĂę read more is the first to combine a barrier nonlinear programming method with a line search filter method, with a fundamental convergence theory for this approach. The result is an efficient, large-scale nonlinear programming code with global convergence properties and fast local convergence properties under weak assumptions. As described in the paper, IPOPT includes a number of refinements that overcome the Maratos effect (slow convergence) and effectively deal with negative curvature and rank deficiency in nonlinear problems. Moreover, it strongly leverages recent advances in automatic differentiation for first and second derivatives along with indefinite factorizations of large, sparse matrices. In independent tests IPOPT has consistently performed among the top state-of-the-art nonlinear programming solvers. IPOPT is available freely, is easy to use, and has interfaces to many modeling packages. As a result, it makes large-scale nonlinear programming accessible to a broad audience.



Wen Wins ICS Student Paper Prize

Zaiwen Wen, a Ph.D. student at Columbia University was the winner of the 2009 ICS Student Paper Award for his work on *A Line Search Multigrid Method for Large-Scale Nonlinear Optimization*, joint with Donald Goldfarb.

The paper presents a line search multigrid method for solving discretized versions of general unconstrained infinite dimensional optimization problems. At each iteration on each level, the algorithm âĂę read morecomputes either a "direct search" direction on the current level or a "recursive search" direction from coarser level models. Introducing a new condition that must be satisfied by a backtracking line search procedure, the "recursive search" direction is guaranteed to be a descent direction. Global convergence is proved under fairly minimal requirements on the minimization method used at all grid levels.

Myths and Counterexamples in Mathematical Programming

James Orlin
M.I.T.

James Orlin is the The Edward Pennell Brooks Professor of Operations Research at the MIT Sloan School of Management. Professor Orlin specializes in network and combinatorial optimization. He has helped develop improved solution methodologies in airline scheduling, railroad scheduling, logistics, network design, telecommunications, inventory control, and marketing. Together with MIT Sloan colleague Thomas L. Magnanti and Ravindra K. Ahuja from the University of Florida, he has written the award-winning text *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, 1993).

The following false statements about topics in math pro-

gramming all have something in common.

1. A transportation problem with unique shipping costs has a unique optimal shipment.
2. In a dynamic lot size problem, a stochastically greater lead time cannot result in a lower average cost.
3. Newton's method (for non-linear programming) converges to a stationary point if the starting point is sufficiently close.
4. In a multi-objective linear program, one should put the greatest weight on the most important objective.
5. Given differentiable functions, an optimal point must satisfy the Lagrange Multiplier Rule.

The common feature of all five false statements is that they are all contained in Harvey Greenberg's collection *Myths and Counterexamples in Mathematical Programming*. The collection is maintained on line at http://glossary.computing.society.informs.org/myths/CurrentVersion/myths.pdf. As Greenberg points out, the counterexamples may be viewed as "erroneous results, paradoxes, fallacies, anomalies, pitfalls, and counterintuitive results." Many of the myths and counterexamples have been claimed as true either in refereed publications or in books.

The collection is currently 133 pages long with over 150 myths and counterexamples. It is divided into sections on the following subtopics: linear programming, integer programming, nonlinear programming, dynamic programming, multiple objective programming, and special form programs.

I am a huge fan of counterexamples as a way of improving mathematical intuition. I find the myths especially interesting. [Truth in advertising: I might not be nearly so enthusiastic about a myth if it was a correction of something I published.]

Greenberg's first version of Myths and Counterexamples was made available to the public in 1996 at the same time that he made available other teaching materials. The idea of myths and counterexamples was inspired to a large extent by counterintuitive results published in the literature. For example, Ron Graham's result that the makespan of a list schedule can increase (get worse) even if the number of processors increases or if the processing time of each job decreases. Graham's counterexamples imply that the computation time of a program running on a computer with parallel processors can increase if the number of processors increases or if the time of each task is decreased.

In its first incarnation, the collection of myths and counterexamples was small and largely unnoticed. Over time, Greenberg gathered more examples. He made a major update to the collection in January, 2008, and another major update in October 2008. The current version was last updated March 15, 2009.

Greenberg has remained especially interested in "more for less" paradoxes such as Graham's makespan paradoxes, and Braess's paradox, which states that adding a new road to a road network can simultaneously increase everyone's transit time.

Greenberg is actively engaged in expanding the collection, and encourages anyone who has a counterintuitive result or fallacy to send it to him. He plans to expand the number of counterexamples on each topic, and leaves open the possibility of adding new topics over time.

The collection will be of interest to anyone who is interested in challenging and improving their intuition in math programming.

ICS Members in the News

Anna Nagurney was a speaker at the 2009 World Science Festival in NYC in mid June. Anna had lofty colleagues as speakers, including five Nobel laureates; mathematicians such as Barrow and Penrose; musicians such as Joshua Bell, Bobby McFerrin, and Yo-Yo Ma; and esteemed academics Alan Alda, Glenn Close, and Harrison Ford. A full list of the speakers is at http://worldsciencefestival.com/speakers/2009.

University of Wisconsin-Madison Computer Scientist **Michael Ferris** was one of five scientists selected to lead work in the new public institute, the Wisconsin Institue of Discovery, to open in December 2010. The goal of the institute is to explore biotechnology, nanotechnology and information technology, said John Wiley, interim director of the institute and former University of Wisconsin chancellor. Tools to improve human health and welfare likely will result, he said. Prof. Ferris won the internal competition with his proposal for a Center for Optimization in Biology and Medicine. The web site for the institute is http://discovery.wisc.edu/.

GPU Computing Meets Optimization

Sangkyun Lee
University of
Wisconsin-Madison

Sangkyun received his M.S. in computer sciences in 2008 in the University of Wisconsin, Madison. He is now a Ph.D. student in the department of computer sciences of the University of Wisconsin, Madison. He is interested in large-scale nonsmooth convex programming problems.

Graphic adapters had been simple devices manufactured for displaying graphics to computer screens. But recent calls for representing realistic and/or realtime graphics have evolved them into massively parallel and programmable units, called graphical processing units (GPUs), which are ready to be used for general computations. The idea of using GPUs for generic computations has been around from late 70's, but it is only recently that the idea became popular, as regular PCs begin to equip powerful GPUs.

GPGPU is the name of performing general computations on GPUs (http://www.gpgpu.org/), and there have been major trend changes of tools for GPGPU:

- OpenGL (2000 ~ present, http://www.opengl.org/): An industrial standard graphics library. It is vender-independent, but primarily designed for graphics, not computations.
- Vendor-dependent platforms (2007 ~ present): CUDA from NVIDIA, CTM from AMD (former ATI). Codes run only on the GPUs from a specific vendor, but more efficiently.
- OpenCL (2009 ~ future, http://www.khronos.org/opencl/): *Open Computing Language*. An open standard for GPGPU, being driven mainly by Apple.

This article focuses on NVIDIA's CUDA (Compute Unified Device Architecture), because it provides an efficient but easy-to-use computing platform akin to the popular C language.

"There is nothing more practical than a good theory (K. Lewin.)"

Albeit this saying accords with my belief, recent developments have opened a new era for practical parallel computing for everybody.

Parallel Computation using GPUs

Parallel computation has a long history, but GPGPU has distinctions from the predecessors.

- GPUs provide many processors optimized for computation at extremely low prices, about two dollars per processor. In 1982, Cray X-MP provided up to four processors at $15 million.
- GPUs are designed to minimize the control overheads of multiple processors. In general multiprocessor environments, user tasks are controlled by operating systems along with other tasks in the system. In CUDA, a device driver, not an OS, manages GPU threads preventing interventions from non-GPU tasks.
- GPUs suffer minimal *context switching* overheads. For a numerical operation, all operands should be loaded into CPU resources named *registers*. As registers are scarce in a CPU, they need to be shared among CPU threads. When a thread A yields a CPU to another thread B, the content of the registers used by A should be stored in the system memory, and the register contents of B should be reloaded to registers from the memory (if any). This context switching suffers from limited memory bandwidth. In contrast, as GPU threads in CUDA receive dedicated set of registers, register loads/unloads are unnecessary.

To emphasize the last point, let's consider parallelizing a 'for' loop in C using threads.

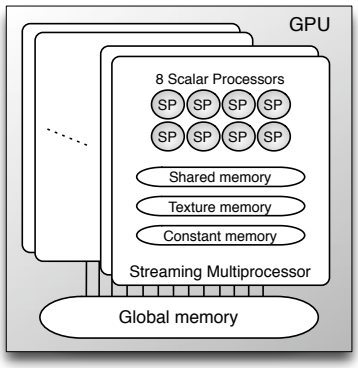


Figure 1. A Schematic view of a GPU. A streaming multiprocessor (SM) has eight scalar processors (SPs), connected to global memory.

```
for(i=0; i<BIG_NUMBER; ++i)  C[i] = A[i] + B[i];
```

It will be a simple but bad idea to spawn a `BIG_NUMBER` of CPU threads, each of them computing one component of the array `C` – as many threads contend for the smaller number of CPUs, context switching overhead will soon surpass the advantage of parallelism. With GPUs, on the other hand, creating many threads is the usual way to parallelize such a loop.

There have been many researches in various areas reporting great speed-ups of numerical tasks using GPUs, including machine learning , medical imaging , DNA sequence alignment , and molecular modeling/simulations (see [1] for references therein.) GPUs have been used for implementing optimization methods like Newton's method [Toledo07] and conjugate gradient method [Bolz03]. GPU-based matrix factorization algorithms [Volkov08,Jung08] will be useful for many other optimization techniques.

NVIDIA GPUs

NVIDIA provides a wide range of GPU products that can be used for GPGPU. While old models like GeForce 8800, 9800 provide only single precision floating point computations, new products like GeForce GTX 280 support double precision as well. Tesla products provides GPUs with larger memory. We list some of the most popular GPUs in Table 1.

All NVIDIA GPUs consist of *streaming multiprocessors* (SMs), each of them is composed of 8 *scalar processors* (SPs). For example, a GeForce GTX 280 in Table 1 has 30 SMs, and

Name	SPs	memory (bandwidth)	precision
GeForce GTX 280	240	1 GB (141.7 GB/s)	s & d
GeForce 9800 GTX	128	512 MB (70.4 GB/s)	s
Tesla C1060	240	4 GB (102 GB/s)	s & d
Tesla C870	128	1.5 GB (76.8 GB/s)	s

Table 1: NVIDIA GPUs. New devices support both single('s') and double('d') precision floating point operations.

therefore $8 \times 30 = 240$ SPs in total. An SM also has one instruction unit, 8192 registers, small on-chip shared memory and read-only caches.

GPUs have off-the-chip memory called *global memory*, which is accessible from all SPs within a single GPU. The connection between the global memory and SPs is very fast (see Table 1); however, global memory and a host computer connects via PCI-Express bus, which is much slower (8 GB/s.)

CUDA Platform

CUDA is a free software platform to support computations on NVIDIA GPUs (http://www.nvidia.com/object/cuda_home.html.) It provides a complier for CUDA codes, which are extensions of the standard C programs. A CUDA code contains CPU parts that call GPU routines with thread execution configurations, and GPU parts called *kernels*.

CUDA splits a computation task into a grid of *blocks*, where a block consists of a set of threads. Each block is automatically scheduled to run on a certain SM, and as there are eight SPs in an SM, eight threads in each block run concurrently at a time. All SMs run concurrently and can be synchronized by *barriers*. All SPs in the same SM run the same instruction, but at possibly different states and with different streams of data. This computation model is called a single-instruction multiple-thread (SIMT) model.

For example, let's see how CUDA adds two N-by-N matrices A and B, and then store the result as the matrix C. The CPU part looks as follows:

```
int main()
{
    dim3 block(16,16);
    dim3 grid((N + block.x - 1) / block.x,
              (N + block.y - 1) / block.y);
    matAdd<<<grid, block>>> (C, A, B);
}
```

In this code, each thread block is defined to have 16×16 threads. We split the matrix dimension into blocks, defining a grid. Finally we call a GPU routine `matAdd`, with the grid and block configurations; CUDA uses a special construct <<< ... >>> for this. The GPU part looks as follows:

```
__global__ void matAdd(float** C, float **A, float **B)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}
```

This GPU function is executed by each thread in the grid. The qualifier `__global__` represents that this function can be called from both CPU and GPU parts. The global variables blockIdx , `blockDim` and `threadIdx` of CUDA tell the location of the current thread in the grid. Each GPU thread computes one output entry of the matrix addition.

For convenience, CUDA provides GPU versions of BLAS Level 1, 2 and 3 operations in CUBLAS library, and discrete

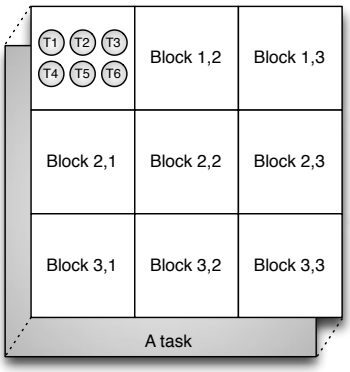


Figure 2. A CUDA task split into a 3-by-3 grid of nine blocks, with six threads in each block.

Fourier transforms in CUFFT library. You can also try Jacket GPU engine for Matlab (http://www.accelereyes.com/) to feel GPU performance without changing your existing Matlab codes too much. These pre-built libraries are easy-to-use, but often slower than customized codes – in order to get the maximum performance out of GPUs, you would have to create GPU routines by yourself.

"You have to learn the rules of the game. And then you have to play better than anyone else (A. Einstein.)"

In [1], algorithms to solve large-scale nonsmooth convex optimization problems, compressive sensing and image restoration, are implemented using native CUDA, finding solutions maximally 100 times faster than the same algorithm implemented in Matlab. Fast implementations are important in both applications: in compressive sensing, the allowance of making fewer observation comes from the assumption that we can solve the reconstruction problem efficiently; fast image restoration is necessary in order to be used as a pre-processing step of realtime tasks, computer vision for example.

Compressive Sensing

The idea of compressive sensing is to reconstruct a *sparse* signal from a few observations. Given that a signal vector $x \in^n$ is *S-sparse*, which means that the number of nonzero components of x is at most S, we aim to recover x from the observations $y \in^m$ with $m \ll n$, where each observation is some linear function of the signal; y can be represented as $y = Ax + z$ with some matrix $A \in C^{m \times n}$ and a noise vector z. We formulate the recovery problem as

$$\min_x \|x\|_1 \text{ subject to } \|Ax - y\| \leq \epsilon. \qquad (1)$$

The theory of compressive sensing tells that if the transformation matrix A satisfies a certain property called *restricted isometry property* (RIP), then the solution of (1) recovers the original signal x. RIP ensures that all distances between any two S-sparse signals are well preserved in the measurement

space, after being operated on by A. And thankfully, simple construction of A is often suffice. For example, we can form A by sampling n columns uniformly at random on the unit sphere of ^{m}. With very high probability, the matrix A acquired by this process obeys RIP provided that

$$m \geq C \cdot S \log(n/S)$$

for some constant C. In other words, (1) yields the true signal from m measurements even when m is only a modest multiple of S. Compressive sensing has many interesting applications, including single-pixel camera, MRI, DNA microarray, etc (see more in http://dsp.rice.edu/cs.)

Separable approximation for compressive sensing

The problem (1) can be reformulated as

$$\min_{x \in^n} \ \phi(x) + \tau \|x\|_1, \tag{2}$$

where $\phi(x) := \frac{1}{2}\|Ax - y\|_2^2$ and for some regularization parameter $\tau > 0$. Note that for $\tau \geq \tau_{\max}$ the solution is $x = 0$, where

$$\tau_{\max} := \|A^T y\|_\infty$$

Although (2) is a simple convex quadratic program, the high dimensionality of x and the fact that A is dense in many applications give rise difficulties.

Among the many algorithms to solve the problem (2), we focus on the SpaRSA approach of [6]. SpaRSA uses a second-order approximation of the smooth part $\phi(x)$ of (2) at the current iterate x^k; SpaRSA obtains the new iterate x^{k+1} by solving the subproblem:

$$x^{k+1} = \arg\min_z \frac{\alpha_k}{2}\|z - x^k\|_2^2 + (z - x^k)^T A^T (Ax^k - y) + \tau\|z\|_1 \tag{3}$$

for some $\alpha_k > 0$. The subproblem (3) is the same as (2) except that the true Hessian $A^T A$ replaced by $\alpha_k I$, and a constant term is omitted. It requires $O(n)$ operations to solve the subproblem as it is separable in the components of z, and for each component a closed-form solution exists. We choose α_k so that $\alpha_k I$ mimics the behavior of the true Hessian along the last two iterates, inspired by Barzilai and Borwein [Barzilai88]. Set α_k by

$$\alpha_k = \frac{(s^k)^T y^k}{(s^k)^T (s^k)} = \frac{\|As^k\|_2^2}{\|s^k\|_2^2} \tag{4}$$

where $s^k := x^k - x^{k-1}$ and $y^k := \nabla\phi(x^k) - \nabla\phi(x^{k-1})$. This approach does not necessarily give a decrease in the objective (2) at each iteration; a monotone variant of SpaRSA uses the value (4) as an initial guess, then repeatedly increases α_k by some constant factor until the new iterate x^{k+1} gives a lower function value than x^k.

SpaRSA on GPUs

$\tau/\tau_{\max}$	CPU		GPU		Speedup	
	iters	time (s)	iters	time (s)	total	iter
0.000100	107	107.08	129	2.08	51	62
0.000033	131	129.10	131	2.10	61	61
0.000010	149	145.31	160	2.57	57	61

Table 2: Computational results for a 1-D DCT sensing matrix of dim 131072×1048576, with 26214 spikes

$\tau/\tau_{\max}$	CPU		GPU		Speedup	
	iters	time (s)	iters	time (s)	total	iter
0.10	64	98.25	64	1.00	98	98
0.05	65	103.10	70	1.08	95	102
0.02	80	117.97	84	1.30	91	95

Table 3: Computational results for a 2-D DCT sensing matrix of dim 20972×1048576, with 1031 spikes

As the major operation in SpaRSA is the matrix-vector products involving A and A^T, it is important for a GPU implementation to store A compactly and compute the multiplication efficiently. We formed our matrix A by randomly selecting m rows from an n-dimensional discrete cosine transform (DCT) matrix. This matrix A satisfies RIP, and the matrix-vector product for A can be calculated in $O(nlogn)$ operations using fast Fourier transform (FFT) algorithms (with $O(n)$ pre- and post-processing steps) . We used the FFT routine provided by CUFFT library.

All the other linear algebra operations can be implemented using CUBLAS library, but often operations are better to be implemented natively in CUDA, rather than split into elementary operations, for better GPU utilization.

We measured the time taken to acquire the solution of (2) using SpaRSA algorithm, comparing the implementation on CPUs (using Matlab with a single core of an Intel quadcore CPU at 2.66GHz) and the implementation on GPUs. For experiments we first constructed a one-dimensional signal of length n, with a sensing matrix consists of $m \ll n$ rows drawn randomly from an $n \times n$ DCT matrix. The signal consists of $\lfloor m/5 \rfloor$ spikes, half of which have magnitude near 1 with the remaining half having magnitude between 10^{-5} and 10^{-4} (logarithms uniformly distributed), with noise of order 10^{-6} on each element. For two-dimensional experiment, we construct a two dimensional signal of length $n = \bar{n} \times \bar{n}$ for some positive integer $\bar{n}$. The signal consists of spikes of $+1$ or -1, of which the fraction is .001. We set m to be 20 times the number of spikes, and measurements are corrupted by noises drawn independently from a normal distribution.

To run the GPU implementation, we use one of the two GPUs in a GeForce 9800 GX2 device, which has 128 scalar processors and 512 MB of global memory (64 GB/s). The results are shown in Table 2 and Table 3. The CPU and GPU algorithms are identical but give different numbers of iterations, as the CPU code uses double precision but the GPU code uses single precision arithmetic. But both implementations return

solutions with almost the same mean square error. For 1-D case we achieve about 60 times of speedup, whereas for the 2-D case up to 100 times.

Image restoration

Given a compact image domain Ω, we want to restore the error-free image $u \in \Omega$ from a distorted image f, which is assumed to be obtained by some linear transform,

$$y = Bu + z \tag{5}$$

where B is a transform matrix and z is a noise component.

The image is restored by solving the total-variation (TV) regularization problem introduced by Rudin, Oscher, and Fatemi [8],

$$\min_u \int_\Omega |\nabla u|_2 + \frac{\lambda}{2}\|y - Bu\|_2^2. \tag{6}$$

This method is known to be highly effected in removing unwanted fine-scale detail while preserving edges. Assuming that u has bounded variation, we can rewrite

$$\int_\Omega |\nabla x| = \max_{\|w\|_2 \le 1} \int_\Omega \nabla u \cdot w = \max_{\|w\|_2 \le 1} \int_\Omega -u\nabla \cdot w. \tag{7}$$

Using (7), we can form a minimax problem of (6) as follows

$$\min_u \max_{\|w\|_2 \le 1} \ell(u, w) := \int_\Omega -u\nabla \cdot w + \frac{\lambda}{2}\|y - Bu\|_2^2. \tag{8}$$

Note that $\ell(u, w)$ is convex in terms of u and concave in terms of w, and the saddle point is attained.

We focus on the primal-dual hybrid gradient projection (PDHG) approach proposed by Zhu and Chan [9]. This method generates a sequence of primal-dual pairs (u^k, w^k) by performing dual ascent and primal descent updates at each iteration,

$$w^{k+1} := P_{\{w:\|w\|_2 \le 1\}}(w^k + \tau_k \nabla_w \ell(u^k, w^k)) \tag{9}$$

$$u^{k+1} := u^k - \sigma_k \nabla_u \ell(u^k, w^{k+1}), \tag{10}$$

where $P_X(v)$ means the projection of v onto the set X. The algorithm repeats these updates until duality gap falls below a certain threshold. The steplengths are defined by $\tau_k := (.2 + .8k)\lambda$ and $\sigma_k := (.5 - 1/(1 + .2k))/\tau_k$ as in [9]. This method is very simple yet has striking performance on practical image restoration problems.

PDHG on GPUs

In GPU implementation, the crucial parts of PDHG are the spatial gradient and the divergence operator appear in the gradient computation of (9) and (10), respectively. In both operations, we split the variables u and w into two dimensional blocks, in which each thread (i.e. each scalar processor) takes care of one component of the output matrix. The SP at the (i, j)-th location of the output matrix has to access not only the (i, j)-th location of the input vector, but also adjacent locations of the input matrix in the i and j directions, which are also needed by other threads. As memory bandwidth is limited, we have to reuse those values if they once read by other threads. A simple but efficient way for reusing is imposing a cache on the input matrix (global memory is not cached by itself.) In CUDA this cache is called as a *texture*, which provides a read-only cache of a user-specified global memory area.

Image size	Tol	CPU		GPU		Speedup	
		iters	time (s)	iters	time (s)	total	iter
128^2	1.e-2	11	0.03	11	0.02	2	2
	1.e-4	79	0.21	79	0.02	11	11
	1.e-6	338	0.90	329	0.07	14	13
256^2	1.e-2	13	0.17	13	0.02	9	9
	1.e-4	68	0.81	68	0.03	32	32
	1.e-6	304	3.57	347	0.11	33	38
512^2	1.e-2	12	0.95	12	0.03	31	31
	1.e-4	54	3.96	54	0.05	76	76
	1.e-6	222	16.08	238	0.19	84	90
1024^2	1.e-2	14	5.42	14	0.08	64	64
	1.e-4	69	25.80	69	0.24	106	106
	1.e-6	296	103.54	324	1.02	102	111
2048^2	1.e-2	13	31.41	13	0.28	114	114
	1.e-4	67	149.24	67	0.90	165	165
	1.e-6	319	694.16	338	4.12	169	179

Table 4: Computational results of image denoising (λ=0.041.)

Table 4 shows the result of image denoising experiments, which corresponds to the case when the matrix B in (5) is an identity matrix. The input images are contaminated by a Gaussian noise of mean 0 and standard deviation 0.1.

GPUs: yet another coprocessors?

In 80's, the Intel processors like 8086, 80286 and 80386 lacked floating point computation (FP) hardware units. As FP computations were done by software, if you wanted to run FP-computation-hungry applications, you had to buy separate hardware called 'coprocessors', which had model names '80x87'. Now it is hard to even hear of coprocessors, because they are behind the scene, embedded even in consumer CPU chips.

GPUs look pretty much like a new kind of coprocessors; now they provide huge FP computation horsepower by means of massive parallelism, or by going multi-cores, which is the virtue of newest computation units. Would someday we see those new coprocessors merge with CPUs? We won't have to wait for long. Intel's new Core 2 CPUs (codename Nehalem) with integrated on-chip GPUs are to be unveiled in Q3 2009. This CPU-GPU integration will relieve the burdens of moving data between CPUs and GPUs, which choke the current generation of GPUs. GPU computing is making impractical practical, giving optimizers the chances to challenge more difficult problems.

References

[1] S. Lee and S. J. Wright. "Implementing algorithms for signal and image reconstruction on graphical processing units," submitted, 2009.

[2] J. H. Jung and D. P. O'Leary, "Implementing an interior point method for linear programs on a CPU-GPU system," *Electronic Transactions on Numerical Analysis (ETNA)*, 28, pp. 174–189, 2008.

[3] V. Volkov and J. Demmel, "LU, QR and Cholesky factorizations using vector capabilities of GPUs," Technical report, UCB/EECS-2008-49, 2008.

[4] J. Bolz, I. Farmer, E. Grinspun, and P. Schr/:oder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," *ACM Transactions on Graphics*, 22, pp. 917–924, 2003.

[5] R. de Toledo, B. Levy, and J-C Paul, "Iterative Methods for Visualization of Implicit Surfaces on GPU," *International Symposium on Visual Computing (ISVC)*, pp. 598–609, 2007.

[6] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, March-April 2008.

[7] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA Journal of Numerical Analysis*, vol. 8, pp. 141-148, 1988.

[8] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259-268, 1992.

[9] M. Zhu and T. F. Chan, "An efficient primal-dual hybrid gradient algorithm for total variation image restoration," Mathematics Department, UCLA, CAM Report 08-34, May 2008.

Introduction to Derivative-Free Optimization, *A. R. Conn, K. Scheinberg, and L. N. Vicente, SIAM, 2009.*

Todd Munson
Mathematics and Computer Science Division Argonne National Laboratory

Todd Munson is a member of Argonne 's Mathematics and Computer Science Division, working primarily in the areas of large-scale continuous optimization and nonlinear complementarity problems. Munson is a lead developer of PATH, the most widely used code for solving complementarity problems; the Network Enabled Optimization System (NEOS), a collaboration between Argonne and Northwestern University that provides access to optimization packages through a variety of Internet interfaces; and the Toolkit for Advanced Optimization (TAO), an open source collection of parallel algorithms for solving large-scale nonlinear optimization problems.

Introduction to Derivative-Free Optimization offers a brief tour through pattern-search and model-based methods for derivative-free optimization and the important research being performed. The book is split into two main parts: the first develops the mathematical framework, and the second develops globally convergent algorithms. A third section briefly mentions other topics, such as the emerging research on derivative-free methods for constrained optimization problems. An appendix has pointers to some available derivative-free optimization software. Readers looking for a discussion of other techniques, such as genetic and evolutionary algorithms and simulated annealing, should look elsewhere since these are not covered in this volume.

The mathematical discussion in the first part of the book provides suitable background for the later algorithmic discussion and includes chapters on sampling, on properties of interpolation and regression models using Lagrange polynomials, and on techniques for computing with Lagrange polynomials. The relationships between Λ-poisedness, the condition number of the interpolation matrix using the shifted, scaled sample set, and the resulting error bounds are particularly useful in understanding the later algorithms and convergence proofs. While several examples are included in the text, these sections may have benefited from additional illustrative examples to give the reader more intuition.

The chapter on computational methods for Lagrange polynomial is also good but could have been expanded. The basic algorithm to compute the Lagrange polynomials for well-poised sample sets via an orthogonalization procedure can be easily coded from the description. The completion and improvement algorithms for ill-poised sets are harder to implement because they may require the use of a global optimization method. The text in the trust-region chapter on finding global solutions for quadratic functions in a ball could have either been included in this chapter when discussing quadratic models or turned into an appendix referenced by both chapters, for example, to provide the reader a more complete algorithm. All the computational methods presented can be sensitive to finite-precision arithmetic. While the resulting interpolation error may not matter in practice, a discussion on the effects of finite-precision arithmetic and mitigation strategies and the consequences of only approximating globally optimal solutions could have been included.

The algorithmic discussion based on this mathematical framework includes pattern-search, Nelder-Mead, line-search, and trust-region methods. Roughly half of this section is devoted to the trust-region framework and algorithms based on the trust-region framework. This material is well developed and offers a good contrast to trust-region methods when derivatives are available. In the latter case, we want the trust-region radius to become large, while in the derivative-free case, the trust-region radius needs to converge to zero, since this is indicative of convergence to a stationary point for fully linear and quadratic models. While this emphasis on trust-region methods is understandable given their importance, an expanded

treatment of other topics, such as the line-search method using simplex derivatives with a limited-memory quasi-Newton Hessian approximation or the extension of the pattern-search method to nonsmooth functions, would have been desired. In particular, much of the algorithmic coverage pertains to fully linear and quadratic models of the objective function, and most convergence proofs assume the objective function is at least continuously differentiable with a Lipschitz continuous gradient. On a few occasions, however, results based on convex analysis are included where the objective function is assumed to be only Lipschitz continuous. The inclusion of these results is necessary to demonstrate applicability of the methods to problems where derivatives do not exist. The book, however, would have benefited from an appendix on convex analysis to familiarize readers with the relevant concepts.

In short, the book is well written, with a good mixture of topics. It is not a complete review of every topic, but that is to be expected given the ongoing activity in this community. I highly recommend this book for researchers wanting to learn about the current state of the art in pattern-search and model-based derivative-free optimization and for those interested in pursuing research in this area. Much of the material could be used in a graduate course on derivative-free optimization but may need to be supplemented depending on the choice of topics.

Editor's Note: An errata for the book is available at http://www.mat.uc.pt/~lnv/idfo.

Clarification of Criteria for Publication: Heuristic Search and Learning

David Woodruff
University of California-Davis

The editorial policy for the Heuristic Search and Learning Area, as given on the web site for the INFORMS Journal on Computing is:

> This area focuses on the application of heuristic methods for solving difficult operations-research problems or in learning contexts. In particular, it covers topics such as Metaheuristics (Tabu Search, Scatter Search, Genetic Algorithms and Evolutionary Methods, Ant Algorithms, Simulated Annealing, etc.) and Neural-Network techniques. Hybrid approaches combining existing heuristic methods, alone or in conjunction with techniques from other areas of operations research or computer science, are also of particular interest. The emphasis within the area is on papers presenting methodological innovations that can be applied to a wide range of problems or situations. Survey papers covering recent advances in a given field and papers aimed at providing a conceptual integration of the area are also welcome.

A sentence of interest is: "The emphasis within the area is on papers presenting methodological innovations that can be applied to a wide range of problems or situations." A simple assertion that the innovations can be applied elsewhere does not meet the burden of proof required in good scientific practice. The authors must demonstrate generality in a convincing manner, either experimentally or theoretically. Rarely, it may be clear that a method is broadly important even when it is tested on only one problem, but this would be very unusual.

For many outlets in our field, a necessary and sufficient condition for publication is to show better results than a set of competitors over a portion of some set of test instances. While this may be a reasonable standard in some settings, it is generally insufficient for the INFORMS Journal on Computing, especially given that a significant fraction of our audience are not specialists in heuristic search and learning. Obtaining some best-known results will be helpful in making the case for the paper, but it is not normally sufficient.

The problems considered in the paper must be "important" in some sense, though this is difficult to define precisely and is subject to some tradeoffs. Importance may be demonstrated by application to a problem of practical significance, or demonstration that the problem has been extensively studied in the research literature, for example.

A scientifically rigorous paper presenting methodological innovations that can be applied to a wide range of important problems or situations is not easy to create. The good news is that it is relatively easy to recognize such a paper, and they will be published quickly because they are very valuable. Papers in this category will not generally require numerous major revisions that need to go back to the referees: a paper presenting methodological innovations that can be applied to a wide range of problems or situations does not need a committee to make it perfect. Conversely, the referees generally canâĂŹt walk authors through the process of discovering and describing methodological innovations that can be applied to a wide range of problems or situations.

Survey papers must provide "conceptual integration", which rules out annotated bibliographies, though these are quite useful and heavily cited. Of course, survey papers that provide conceptual integration are even more useful, and when one of those is submitted we will work to review it and accept it quickly so that it appears in a timely manner.

Message from the Chair

(continued from page 1 ◁)

Treasury: At the beginning of 2010, the Society had approximately $22,935.72

Membership: We started 2010 with 461 members, of which more than 100 were new members who joined under our successful student membership drive. This year, we created a new Membership Committee led by Dick Barr to continue exploring new means of enhancing the society's value to its members and growing our ranks. (Renew your membership!)

Presence at National Meeting: In San Diego, we had 56 sessions, a bit of a decrease from last year but, given the overall state of the economy, a respectable showing. It looks like ICS will also have a significant presence at this years meeting in Austin, Texas.

ICS Conference: We are preparing for our 12th (!) conference to be held January 9–11, 2011 in beautiful Monterey, CA. For some reason it took the folks at INFORMS a bit longer than usual to negotiate and finalize the selection of venue and so we are getting a late start. But, we are beyond that now. Please mark your calendars. Submission deadline for abstracts is October 1, 2010.

Prizes: At the Austin meeting, we will give out two prizes: the ICS Prize and the ICS Student Paper Award. Also, at the Monterey meeting we will be awarding the second Harvey J. Greenberg Award for Service to ICS, which acknowledges the life-time achievements of people who had an impact on ICS. There is no shortage of deserving candidates for any of the awards, and this year it is even easier to apply using the new standard ICS prize email addresses. For details on selection criteria and nomination procedures, go to the ICS website, http://computing.society.informs.org, and click on *Prizes*.

Thanks to Editor Jeff Linderoth and the team of contributing authors, you can find details on these and all the other ICS happenings in this wonderfully comprehensive newsletter.

This edition marks my first as ICS Chair. I would like to express my heart-felt thanks to outgoing Chair, Robin Lougee, for the wonderful job she did over the past two years. She made ICS both interesting and *fun*.

Thank you to all the outgoing officers, Robin Lougee (past-Chair), Rob Dell (past-Board Member), Pascal Van Hentenryck (past-Board Member), Steve Dirkse (past-Board Member), Matt Saltzman (past-Board Member),

We're lucky to have a strong new group of officers at the helm: Bill Cook (Chair-elect), Kipp Martin (Secretary/ Treasurer), in addition to the continuing Board Members: Jonathan Eckstein (Board), and Jonathan Owen (Board). I look forward to working with them all in my new role as your Chair, and with YOU.

ICS is a great opportunity to get involved, meet some new people, and generally combine your professional interests with a little fun. So, what do you need from ICS? What do you want? Let's hear it.

Our next business meeting will be this November at the INFORMS Meeting in Austin, Texas. Come, bring a colleague, bring a student, and bring your ideas. Hope to see you in Austin.

Acknowledgments

The Editor thanks all contributors and those who provided service to ICS. Last, but not least, thanks to Harvey Greenberg.

INFORMS Computing Society

Forming the Building Blocks of Operations Research