ICS Newsletter

Volume 19, Number 2 Fall 1998

Contents

A Science in Trouble

Asim Roy
Arizona State University

Can a whole body of science simply unravel when confronted by a few simple challenges? Can a large body of scientists overlook some very simple facts for a long period of time? From the current debate on how the brain learns, the answer appears to be "yes" for at least one body of science – artificial neural networks or connectionism. A sampling of some recent comments might be a good indicator of this. The first open and public admission that much of the existing science is wrong came from Christoph von der Malsburg, a German neurobiologist and computer scientist affiliated with both the Ruhr University of Germany and the University of Southern California. In commenting on the challenge I posed, which claimed that neural networks do not embody brain-like learning, he remarked, "*I strongly believe that the current paradigm of neural network learning misses very essential aspects of the learning problem, and I totally concur with the assessment, expressed in your expose, that specific prior knowledge is required as a basis for learning from a given domain...I am glad the issue seems to start picking up momentum. Some Kuhnian revolution is required here, and as he (T. Kuhn) wrote, such scientific revolutions are always*

trouble: Continued on page 10

informs Computing Society

The Institute for Operations Research and the Management Sciences' Computing Society Newsletter

Volume 19, Number 2, Fall 1998

ICS Officers

Chair

Harlan P. Crowder (crowder@ilog.com)
ILOG, Inc.
Silicon Valley,, CA

Chair-Elect

Ramayya Krishnan (rk2x+@andrew.cmu.edu)
Carnegie Mellon University
Heinz School of Public Policy & Mgmt
Pittsburgh, PA 15213
(412) 268-2174

Secretary/Treasurer

Bjarni Kristjannson
(bjarni@maximal-usa.com)
Maximal Software
Arlington, VA 22201
(703) 522-7900

Board of Directors

Andrew Boyd (boyd@prosx.com)
PROS Strategic Solutions, Inc.
term expires in 1999

Bruce Golden (bgolden@umdacc.umd.edu)
University of Maryland
term expires in 2001

Christopher Jones (cvj@u.washington.edu)
University of Washington
term expires in 1999

Jeffery L. Kennington (jlk@seas.smu.edu)
Southern Methodist University
term expires in 2001

Matthew J. Saltzman (mjs@clemson.edu)
Clemson University
term expires in 2000

Carol Tretkoff (tretkoff@ilog.com)
ILOG, Inc.
term expires in 2000

Views expressed herein do not constitute endorsement by INFORMS, the INFORMS Computing Society or the Newsletter editors.

Newsletter Staff

Co-Editors

S. Raghavan (suraghav@rhsmith.umd.edu)
Decision & Information Technologies Dept.
Robert H. Smith School of Business
University of Maryland
College Park, MD 20742
(301) 405-6139

Tom Wiggen (wiggen@acm.org)
Dept. of Computer Science
University of North Dakota
Grand Forks, ND 58202-9015
(701) 777-3477

Associate Editors

John N. Hooker (jh38+@andrew.cmu.edu)
Carnegie-Mellon University
Graduate School of Industrial Administration
Pittsburgh, PA 15213
(412) 268-3584

This position is available

Contact the editors

The ICS Newsletter is published semiannually on April 1st and October 1st by the INFORMS Computing Society (ICS). Manuscripts, news articles, camera-ready advertisement and correspondence should be addressed to the Editor. Manuscripts submitted for publication will be reviewed and should be received by the Editor three months prior to the publication date. Requests for ICS membership information, orders for back issues of this Newsletter, and address changes should be addressed to:

INFORMS Computing Society Business Office
901 Elkridge Landing Road, Suite 400
Linthicum, MD 21090-2909
phone: (410) 850-0300

Message from the Editors

S. Raghavan (U of Maryland)
Tom Wiggen (U of North Dakota)

This month, we are pleased to feature two intriguing articles by Asim Roy of Arizona State Univerity and Greg Astfalk of Hewlett-Packard.

Asim tries to expose some of the shortcomings of neural-nets as they are presently implemented. A related article by Professor Roy will appear in the December issue of OR/MS Today, and he plans to offer a tutorial at the ICS Conference in Cancún in January 2000.

Greg Astfalk writes about the present state-of-the-art in our exploitation of parallelism. Greg concludes that this is not yet ready for prime-time.

We are also pleased to include a news item about the presentation of the von Neumann prize to Fred Glover. As you may recall, Fred wrote the feature article for the last issue of this newsletter.

As newsletter editors, we are continually on the lookout for regular columns that the ICS membership will enjoy. One example of this is the "Member Profile" series which comes about through the initiative and efforts of associate editor John Hooker. There must be a number of candidates for other regular columns that you would enjoy reading (web-based applications, web-based teaching and/or learning, views from industry, etc.), and there may be some enterprising individuals among our readers who would enjoy producing columns in those special areas for the ICS newsletter. If you have an idea for a regular newsletter column or a special column, please let us know.

The front cover of this newsletter does not contain any official ICS identifying marks. As INFORMS' newest society, we are at present without official logos of any sort. The society's officers are working to remedy this situation. Relay your suggestions to an ICS officer.

Finally, we welcome your letters and emails to the editors containing news notes, opinions or anything else you want to tell us.

What about parallel programming and high-end systems?

Greg Astfalk
Hewlett-Packard Company
3000 Waterview Pkwy.
Richardson, TX 75080-1400
astfalk@rsn.hp.com

We are going to talk about parallel computing and high-end systems. Parallel processing does not need any introduction, nor definition. Despite this we offer one; to invoke *concurrent* computations for portions of a single application. Key are the words "concurrent" and "single application." There are many important applications that are so blatantly parallel that they are not of concern or interest to us here. Also not of interest here is the notion of the concurrent throughput of many individual jobs. Parallel processing is often thought of as a new topic; it isn't. Parallelism has been around for quite some time, yet despite this we can not call it a mature technology.

High-end systems may need a bit of explaining. The issue is that it means different things to different people. Someone might, legitimately, consider a cluster of PC's (e.g., a Beowoulf cluster) as a high-end system. Others might say that it is a classic Cray supercomputer architecture. Both can be correct depending on the application and numerous other factors, some of which are fiercely "religious" in nature.

Prime-time is that temporal window when most people are watching television. The key is large numbers relative to any other time. With a view toward parallel processing, its prime-time will be when large numbers of people are utilizing it. Today this is, without debate, not the case. Considering the universal set of computer users and programmers it is a minority that use it, understand it, have the time, charter and talent to invoke it, or can benefit from it. What is holding it back? Two things. Ease of use and affordability.

Owing to lack of space we are brief and thus make

only what we feel are the most important points. To do either topic complete justice would require a tome of epic proportions. By its nature this note may be considered subjective, despite our attempt at an objective view. We will address these two points from a *very* pragmatic perspective. Naturally since this note isn't authored via a consensus vote it will be viewed either favorably or as heresy and BS. This author claims full responsibility either way. Some might say this note is cynical, we respond by saying it is honest.

The prerequisites

If the reader will permit the somewhat parochial perspective we will focus on all endeavors in this field, with the exception of research. Our justification is that research is by its nature expected to take time, hard work, false starts, etc. It is by research that the state of the art is advanced. In this note we are considering the practical application of parallel computing.

To actually do parallel computing there are three prerequisites; (1) the need, (2) the machine, and (3) the charter and the time. The need is that the user is faced with a problem that is large enough and important enough that it can justify the expenses and efforts of items (2) and (3). A problem that can be accomplished in a reasonable time on a workstation or server is not likely a candidate. What constitutes a "reasonable" time means different things to different people.

To do parallel computation requires a parallel machine. Thus the user's organization either has to have such a system, be compelled enough by the intended parallel applications to buy one, or to buy time on one outside the organization. Subtle and implicit in this is that the job (i.e., the need) must have some portion of, or the entire, system to itself to actually run in parallel (there are exceptions to this --gang-scheduling-- but this is beyond the scope of this note). This reinforces the "need" part. The application must warrant having a rather expensive computing resource to itself.

The charter and time are necessary since the user's management must allow the programmers and developers the time to make the code run in parallel. It is not uncommon to require efforts of 6-12 person-months in order to get a fully functional parallel code. An exception to this point is that the practitioner could simply use a pre-existing ISV (independent software vendor) code that has already been parallelized and solves the problem at hand. Then item (3) is not relevant.

So what is different in the context of parallel programming? It is a *fact* that parallel programming is (much?) more difficult than sequential programming. Thus it will take us longer to code the application so it is important that the application in question warrants the extra effort. The justification could be the sheer importance of the code to the funding (i.e., developing) organization or the sheer size of the computation that requires going to the extra effort to get the time-to-solution down as low as possible.

When an organization buys a computing resource the inherent expectation is that it will enable better, or more timely, products and product development times. This applies to database systems, high-end technical servers, or desktop NT systems. Another characteristic of virtually all organizations that have large IT budgets for computing systems is that they have a *community* of users. Should the multiple processors of a parallel system be used to serve multiple users, or many independent tasks, in throughput mode. If the parallel system is dedicated to a single application for concurrent solution then there are two compounding factors. First is that the *p* processors being used concurrently are denied to the *p*, or more, users that could be working on their applications on the same system. Thus parallel computations represent a lost opportunity cost to the organization unless the parallel applications' benefit outweighs it. The compounding factor is that parallel applications never fully utilize all of the processors that they are "using." We discuss this in the next section.

From the current President's Information Technology Advisory Committee (PITAC), "there is substantive evidence that current scalable parallel architectures are not well suited for a

number of important applications, especially those where the computations are highly irregular or those where huge quantities of data must be transferred from memory to support the calculation." At least the reader will realize it isn't just this author that has a less than optimistic view of parallelism.

Measuring parallel programs

In the field of parallel programming there are any number of measures of the parallel application; number of processors used, sustained flops, speedup, efficiency, isoefficiency, etc. While these measures do tell us something, they are all off the mark regarding what really matters. The only important measure is that the time-to-solution is less when the application is run in parallel. We should also point out that time-to-solution ought to mean from the time the complete application is started until it is finished. Achieving, and reporting on, linear speed-up on a kernel which represents only a portion of the application's total time is misleading to the ultimate end-users perspective. Failing on this measure may hit the mark in a research setting but not in the real-world.

Let's be honest about what performance we can expect. Without knowing anything about the application or coding this author makes the assertion that on a contemporary RISC processor the application will get 10% of peak performance in sustained, sequential execution. Be advised that the 3-σ on this number is large. When parallel execution is invoked the assertion, again by this author, is that you lose 50%. Specifically, if you sum the 10% of peak over the p processors involved in the computation and then divide that by 2 you get a guesstimate on what will be delivered from the parallel application. The 50% efficiency is for a moderate number of processors, say 8-16. It invariably gets lower with increasing p. We know of cases where only 1% (no typo, we mean one (1) percent) of the aggregate performance of the system is realized. This is, by any perspective, disappointing. For emphasis, we say again that these numbers can vary substantially depending on the application, algorithm, and coding.

Parallel programming

What is the "state of the art" in parallel programming today? It involves significant neurons. The actual effort of producing a parallel code has a significant "human in the loop" component. It would be nice and highly useful if there were a way to automagically produce efficient parallel code; Be advised; there isn't!

We have worked on parallel computing for a long time and where exactly are we today? It *still* requires careful programming, careful attention paid to data locality and control decomposition, and a rather thorough understanding of computer architectures. This is further complicated by the many parallel architectures that are in use today.

If you step back far enough from the parallel programming issue you have the sense that it is practiced the same as it was 15 years ago. The primary method is explicit decomposition of the application (data or control) via message-passing. It is certainly true that MPI, and its forerunner PVM, are much better in their syntax and functionality than the old vendor specific messaging libraries. However, from the programmer's perspective the task is all too familiar.

What MPI has done is significant it that it offers the greatest common denominator for parallel programming. MPI's portability is significant in that a MPI application can run on any type of architecture in use today. This includes SMPs, ccNUMA systems, clusters, Internet-based systems, etc. To be a bit more quantitative, we have data that for this author's company's user population something like 50-60% of the applications are MPI-based. This includes a collection of systems that are mostly SMP architectures.

Reports are available that show efforts in the 6-12 person-month range to parallelize a code. This is done by smart people. The net result for this effort is a far too large disparity between the theoretical peak performance of the system and the performance that the application delivers.

The difficulty lies in the ability of compilers to automatically parallelize legacy code. Much research has been done in this area and there is substantial progress. To the average practitioner this technology has shown little benefit. If a new software project is being undertaken there are now quite a few explicitly parallel high-level languages that could be used. Using these languages avoids the need for message-passing. The portability, longevity, and support of these languages has restricted their use to researchers willing and able to endure the shortcomings. The real test of a new language's acceptance and success is when multiple vendors offer it with support, future releases, documentation, etc. Then, and likely only then, will the independent software vendors (ISVs) consider using it. Until that time it is not prime-time for these languages.

There exists a tension between the computation's decomposition (data and control) and the architecture's forte. At the end of the day it isn't uncommon to find that the parallel computation efforts are more troubling than the efforts associated with the original physics. Have we turned physics into difficult programming problems?

High-end hardware

Technology has driven the hardware used in the parallel computing arena through several phases. The good news is that there are indications that a convergence is taking place in the parallel hardware. Despite this there are still multiple choices and this naturally leads to several ways to program parallel applications.

Machines today are hierarchical. We start from the single processor and then move to connected sets of processors that share memory; the SMP. Considering the SMP as a "node" then the nodes can be tightly integrated into a ccNUMA system. In a ccNUMA system any processor in any node can directly address any byte of memory in any other node by processor issued load/store instructions. That is to say the tightly integrated set of nodes has the look and feel of a SMP, albeit with different latencies and bandwidths to some portions of the memory.

The individual SMPs could also be connected is a somewhat looser fashion by, for example, Myrinet. Here the Myrinet connection allows for nearly direct connection of memory in an individual SMP to that in another. While the latency and bandwidth are not nearly so good as for the ccNUMA systems, it is quite respectable.

After this level of hierarchy we have true clusters of SMPs, ccNUMA systems or Myrinet-ed systems. These are connected by standard interconnects such as Ethernet, FDDI, HiPPI, or any other standard interconnect. The distinction here is that there are multiple autonomous operating systems. The resultant is that the global view of memory and the i/o space is lost.

Hardware is converging to the 'clustering' of SMPs. It is with reluctance that we use the word clustering since this term is overloaded in the industry. If you look at the next hierarchical level in hardware above the individual processor it is the SMP. The size of the SMP varies from 2 to 32. The processors constituting the SMP vary from Pentium Pros to almost Gflops PA-RISC or other, processors. The important trait of the SMP is that it offers shared memory. While this sounds good at first there are issues that aren't all that spectacular.

The days of custom designed (for the technical only market) systems are gone and will unlikely ever return. Bandwidth "costs" and if the price of the resulting system is too high it won't sell in large enough volume. Vendors must leverage the volume commodity market space and products in order to survive. This dictates the use of RISC processors. Despite the somewhat less desirable characteristics of RISC processors relative to classic supercomputers, users will need to make this transition in order to use the high-end systems of the future. Future high-end systems *will* be commodity processor based.

High-end vendor economics

At this point we leave the purely technical confines of the preceding sections and discuss what may well be the more important perspective. The economics in the computer industry. It is this author's opinion that the high-end systems

marketplace is largely driven by business realities.

Rick Belluzzio, SGI's Chairman and CEO, has stated SGI is, "currently redefining its supercomputing strategy by folding the traditional vector technologies into its Origin server lines because the current supercomputing model 'is bankrupt and it doesn't work.' " This is a strong statement coming from the company that started, defined, and leads the supercomputing industry.

The world-wide high-end server market is approximately 2 billion dollars a year. This is for servers selling at 1 million dollars and higher. In the absence of any competition a vendor can thus build a 2 billion dollar business. A vendor that is fiscally healthy can spend about 10% of its annual revenue on research and development. Is this enough? The answer is, "no." The cost to develop a high-end system can exceed this amount. A rather extreme example is Cray Computer Company. It spent 350 million to develop its offering. Making it even more difficult is the constraint that a high-end vendor really needs to have more than one design team in place. This is required since the time to develop a system is longer than the effective market life of a system.

The solution to this is what is practiced by all the large vendors today; have alternate sources of revenue other than just the high-end technical. Systems are designed out of more and more commodity parts and the design is targeted to serve, for example, the technical and commercial markets. This provides a larger base to earn revenue in order to sustain the development of systems.

How this came to pass is a confluence of a number of factors. Among them are (1) the increasingly expensive development costs of high-end systems, (2) the shrinking high-end market space, (3) the lack of clear differentiation between commodity computers and true supercomputers, and (4) the convergence between the technical and "commercial" spaces.

On item (3) we make the point that when the Cray 1 was first introduced it was substantially better than the status quo systems used. Specifically 160 versus 4 Mflops and 12.5 versus 100 nanosecond latency. Today it is not unusual to find that a high-end commodity workstation can outperform a high-end classic supercomputer (on some applications). Item (4) is typified by the emergence of a new and broader usage model in the commercial arena. Commercial companies are now doing decision support, financial modeling, fraud detection, data mining, and other applications. These often stress a system in ways that are similar to technical computing rather than the classic transaction processing.

Epilogue

So is parallel programming ready for prime-time today? No. Is parallel programming ready for the masses? No. Is parallel programming going to become so easy that it will be usable by the masses? Not for years. Is parallel computation important? Yes. Is there a future for parallel processing? Definitely!

The advances that are required to make the preceding paragraph more optimistic are almost exclusively in algorithms and software. The levels of spending and effort are not in step with the importance of these areas. Parallel hardware while it is still maturing, developing, and converging is further along in maturity than is the parallel software.

The biggest issue that is inhibiting the main-streaming of parallelism is its inherent difficulty, combined with the mixed results. After significant hard work by talented people some applications show impressive performance. This statement needs to read, "After programming by the average programmer almost any application can achieve good parallel performance." Alas this is some time in the future. When it is a true statement then parallel procesing's prime-time will have arrived.

We do look forward to the wider acceptance of parallelism. It will be interesting to participate in the unfolding of the plot of this on-going saga.

Greg Astfalk is the Chief Scientist of Hewlett-Packard's High Performance Systems Division in Richardson, TX.

ICS Member Profile: Richard E. Rosenthal

I was very fortunate to get introduced to operations research as an undergraduate at Johns Hopkins University. John Liebman was the first to tell me about OR and Eliezer Naddor taught my first class in it, using Harvey Wagner's book. After two weeks of Naddor's great lectures and Wagner's fun problems, I was hooked. The next year, I had a two-semester course in network flows and graphs with Manny Bellmore, using Ford and Fulkerson and lots of classic papers from the literature. Coming from the first generation of my family to go to college, I had absolutely no idea of what being an academic was all about. Those three professors influenced me more than they could have imagined. Bellmore's course introduced me to the idea of research and it led me to choose optimization as a specialty."

That's how Rick Rosenthal describes his start in OR. He graduated from Johns Hopkins in 1972 and then completed a Ph.D. at Georgia Tech in 1975. His first faculty position was in management science at the University of Tennessee. Then in 1984, he went for a one-year visit to the Naval Postgraduate School at the invitation of Jerry Brown and has been there ever since. He became Editor in Chief of *Naval Research Logistics* in 1988. A year and a half ago, he took over as Chairman of the NPS Operations Research Department, one of the largest, oldest, and, according to NPS Dean Peter Purdue, one of the strongest OR programs in the U.S.

With over 40 faculty, close to 200 graduate students, and $2.8 million in 1998 external research funding, managing the OR department takes a lot of time. But Rick says he is counting on avoiding administration "as a life sentence," so he is keeping a strong interest and involvement in research.

He never forgot the first lesson of Professor Naddor's class, which was that all the techniques of OR exist for the purpose of solving real problems. The first real problem Rick worked on was a consulting job as a graduate student. "It was offered to all the professors first, but they thought the client was a flake and I wouldn't have known any better," he says. The job did not turn out successfully and this was a defining incident in his career. The problem involved a furniture factory that the client had recently purchased. Through years of mismanagement, it had accumulated extremely unbalanced parts inventories. "Imagine a plant with a week's supply of table tops and twenty year's worth of table legs. To make matters worse, the finished products are not particularly profitable." The client wanted to know whether to buy more table tops or to scrap the table legs or some combination of the two. There were 100 products facing this dilemma, and he wanted answers immediately.

"I went home after the first meeting and designed a beautiful linear program. Beaming with pride, I went into the Georgia Tech computer center the next day, opened a private account, and asked for access to an LP solver. After all my courses on math programming algorithms at Hopkins and Tech, how hard could it be to use someone else's software implementation of what I thought I understood so well? I figured it would take less than an hour to master the software, so I made an appointment with the client to meet a little later and told him to bring the data. I promised not to go home before presenting him with the optimal solution."

The only LP software available was LP1108, running on the Univac 1108. "The manual was twice the size of the Manhattan phone book, and it required a lot of experience in Job Control Language, which I had never heard of. Suffice it to say, I was not prepared for my meeting and did not get that beautiful LP formulation even close to running that day. No one had ever taught or even mentioned to me what was involved in creating a

matrix generator for an LP."

As a footnote of the story, credit must be given to Bruce Schmeiser, a fellow Georgia Tech graduate student, who had excellent computing skills and industrial experience. "Bruce was already on his way to becoming a famous name in simulation and couldn't care less about LP. He rigged up a clever little FORTRAN program that saved my neck."

The result of this humbling experience was that Rick's research and teaching in optimization have always contained an emphasis on implementation. It also explains his joy and excitement when, first, conversational solvers like LINDO came along and then algebraic modeling languages like AMPL and GAMS emerged. Rick started teaching a 4-day short course on optimization modeling using GAMS in 1988, a course he still offers twice a year. Some companies send people every year to the class. Past students often provide example models or come back as guest speakers. "I love to demonstrate to people with real problems how algebraic modeling languages make it possible to experiment with intricate mathematical programming approaches very quickly and effectively."

Rick's optimization research began with algorithmic developments. "I have a few favorites from those days. The paper on nonlinear networks applied to hydropower [1] got a lot of attention in the water resources community. My work with Steve Brady on an interactive computer graphical solution to a location problem [2] does not read too badly after all these years. Terry Harrison's dissertation on multi-objective optimization applied to forestry [3] helped me learn about both fields. Another pet project related to my favorite result from Bellmore's networks class: the basis-tree theorem for pure network flows. Thanks to the work of my then-future NPS colleagues Jerry Brown and Gordon Bradley, among others, everyone knew that this theorem lets you do fantastically efficient computing with labeling algorithms on the basis-tree instead of linear algebra on the basis matrix. In a paper that probably no more than three people have read, I showed there exists another tree corresponding to the basis-inverse and you can implement the same algorithms just as well with this *inverse-tree* [4]."

Current research is focused on military applications of optimization, jointly with students and colleagues at the Naval Postgraduate School, such as [5]. "Our students are very special. They have already figured out what they want to be when they grow up, and have come to us after already proving they are first-rate performers at their chosen professions. They enable the faculty to develop expertise in military problem areas to which OR and computers can be applied." One recent example of student research supervised by Professor Rosenthal is Navy Lieutenant Scott Kuykendall's thesis [6], which grew directly out of a situation that had frustrated the lieutenant several times in his previous assignment as the Tomahawk strike officer on an Aegis cruiser. Lt.Col. Steve Baker of the Air Force recently completed a Ph.D. dissertation [7], which is a finalist for the INFORMS George B. Dantzig Prize. With Professors Laura Melody Williams and Rosenthal of NPS, and David Morton of the University of Texas at Austin, Baker developed an airlift optimization model that has been used by the Air Force to answer questions about aircraft fleet selection, airfield infrastructure investment, and airlift concepts of operation.

Rick Rosenthal summarizes his career: "I have been extremely lucky to have found OR at an early age, and to have worked with great teachers, colleagues and students all along the way. Dr. Naddor passed away several years ago, but I think he would be very happy to see the applications focus of my work."

[1]. Rosenthal, Richard E., "A Nonlinear Network Flow Algorithm for Maximization of Benefits in a Hydroelectric Power System," *Operations Research*, Vol. 29, 763-786 (1981).
[2]. Brady, Stephen D. and Richard E. Rosenthal, "Interactive Computer Graphical Solutions of Constrained Minimax Location Problems," *AIIE Transactions*, Vol. 12, 241-248 (1980).
[3]. Harrison, Terry P. and Richard E. Rosenthal, "An Implicit/Explicit Approach to Multiobjective Optimization with an Application to Forest Management Planning," *Decision Sciences*, Vol. 19, 190-210 (1988).
[4]. Rosenthal, Richard E., "Representing Inverses in Pure Network Flow Optimization," *European Journal of Operational Research*, Vol. 23, 356-366, (1986).
[5]. Rosenthal, Richard E. and William J. Walsh, "Optimizing Flight Operations for an Aircraft Carrier in Transit," *Operations Research*, Vol. 44, 305-312, (1996).
[6]. Kuykendall, Scott D., "Optimizing Selection of Tomahawk Cruise Missiles," MS thesis in Operations Research, Naval Postgraduate School, March 1998.
[7]. Baker, Steven F., "A Cascade Approach for Staircase Linear Programs with an Application to Air Force Mobility Optimization," Ph.D. dissertation in Operations Research, Naval Postgraduate School, June 1997.

ICS member profiles are intended to keep us up to date on what some of our members are doing. Profiled persons come from academia, industry, military and civilian government.

trouble: continued from page 1

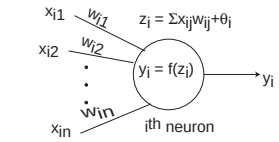
preceded by rising unrest in the community.." And Malsburg is one of the founders of this field. Another founder of this field and a past president of the International Neural Network Society (INNS) confided to me that "*the neuro-boom is over*." But many other scholars have kept on fighting the arguments against the current science on brain-like learning. Another founder of the field and a past president of INNS publicly disagreed with me at the recent debate in Alaska, saying: "*In brief, I disagree with everything he (Asim Roy) said*."

Many distinguished scholars have participated in the two open, public debates at the last two international conferences on neural networks. These debates centered on various aspects of brain-like learning as discussed later in this article. The first debate at the International Conference on Neural Networks (ICNN'97) in Houston, Texas in June, 1997, included four past presidents of INNS and five of the plenary speakers. The second debate at the World Congress on Computational Intelligence (WCCI'98) in Anchorage, Alaska in May, 1998, included five past presidents of INNS, founders and gurus of the field. A summary of the first debate has been published in the INNS Newsletter of May, 1998, and on the Internet through the various neural network-related mailing lists. A summary of the second debate is under preparation.

The debate about connectionism is nothing new. The argument between the symbol system hypothesis of artificial intelligence and the massively parallel system conjecture of artificial neural networks or connectionism has still not abated. Marvin Minsky of MIT characterized connectionism as "naïve" at the first international conference on neural networks in San Diego in 1988. And Minsky and Seymour Papert not only showed the limitations of the earlier simple neural networks, the perceptrons, but were also the first ones to raise the deeper question of computational complexity of learning algorithms ("Epilogue: The New Connectionism" in [8]). But the neural network field moved along heedlessly with its research agenda, ignoring all the deeper and more disturbing questions raised by thoughtful critics. However, a scientific field is destined to stumble sooner or later when it tries to skirt legitimate questions about its founding ideas. Now faced with fundamental challenges to the assumptions behind their brain-like learning algorithms, prominent researchers in the field are finally calling for a "*shake up of the field of neural networks*" and for its "*rebirth*."

Some background information on artificial neural networks

Connectionism or artificial neural networks is the field of science that tries to replicate brain-like computing. The brain is understood to use a parallel computing mechanism where each computing element (a neuron or brain cell in the terminology of this science) in this massively parallel system is envisioned to perform a very simple computation, such as $y_i = f(z_i)$, where z_i is assumed to be a real valued input, y_i is either a binary or a real valued output of the i^{th} neuron, and f a nonlinear function (see Figure 1). The nonlinear function f, also called a node function, takes different forms in different models of the neuron; a typical choice for the node function is a step function or a sigmoid function. The neurons get their input signals from other neurons or from external sources such as various organs of the body like the eyes, the ears and the nose. The output signal from a



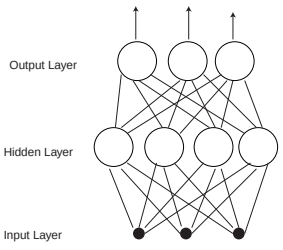


Figure 1

neuron may be sent to other neurons or to another organ of the body.

Studies in neuroscience and neurobiology show that different parts of the brain perform different tasks such as storage of short or long term memory, language comprehension, object recognition and so on. A particular task is performed by a particular network of cells (hence the term neural networks) designed and trained for that task through the process of learning or memorization. These networks, when invoked to perform a particular task, then send their outputs to other parts of the brain or to an organ of the body.

A network can have many layers of neurons, where the outputs of one layer of neurons become the inputs to the next layer of neurons. And a network can have more than one output signal; thus the output layer can have more than one neuron. Different neural network models assume different modes of operation for the network, depending somewhat on the function to be performed. A neural network model for pattern classification is often conceived to be a feedforward type network where the input signals are propagated through different layers of the network to produce outputs at the output layer. On the other hand, a neural network model for memory is often conceived to be of the feedback type (also called recurrent networks or nonlinear dynamical systems) where the outputs of the network are fed back to the network as inputs. This process of feedback continues until the network converges to a stable set of output values or continuously cycles among a fixed set of output values.

Let $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{in})$ be the vector of input signals to the i^{th} neuron, the inputs signals being from other neurons in the network or from external sources. Neural network models assume that each input signal x_{ij} to i^{th} neuron is "weighted" by the strength of the i^{th} neuron's connection to the j^{th} source, w_{ij}. The weighted inputs, $w_{ij}x_{ij}$, are then summed to form the actual input z_i to the node function *f* at the i^{th} neuron: $z_i = \Sigma\, w_{ij}x_{ij} + \theta_i$, where θ_i is a constant, called the threshold value. As mentioned before, some typical node functions are (1) the step function, where $f(z_i) = 1$ if $z_i \geq 0$, and $f(z_i) = 0$ otherwise, and (2) the sigmoid function, where $f(z_i) = 1/(1 + e^{-z_i})$.

A network of neurons is made to perform a certain task (memory, classification and so on) by designing and training an appropriate network through the process of learning or memorization. The **design** of a network involves determining (a) the number of layers to use, (b) the number of neurons to use in each layer, (c) the connectivity pattern between the layers and neurons, (d) the node function to use at each neuron, and (e) the mode of operation of the network (e.g. feedback vs. feedforward). The **training** of a network involves determining the connection weights $[w_{ij}]$ and the threshold values $[\theta_i]$ from a set of training examples. For some learning algorithms like back-propagation [14,15], the design of the network is provided by the user or some other external source. For other algorithms like Adaptive Resonance Theory (ART) [5], reduced coulomb energy (RCE) [10], and radial basis function (RBF) networks [9], the design of the network is accomplished by the algorithm itself, although other parameter values have to be supplied to the algorithm on a trial and error basis to perform the design task.

The **training** of a network is accomplished by adjusting the connection weights $[w_{ij}]$ by means of a ***local learning law***. A local learning law is a means of gradually changing the connection weights by an amount Δw_{ij} after observing each training example. A learning law is based on the general idea that a network is supposed to perform a certain task and that the weights

have to be set such that the error in the performance of that task is minimized. A learning law is **local** because it is conceived that the individual neurons in the network are the ones making the changes to their connection weights or connection strengths, based on the error in their performance. Local learning laws are a direct descendent of the idea that the cells or neurons in a brain are autonomous learners. The idea of "autonomous learners" is derived, in turn, from the notion that there is no homunculus or "a little man" inside the brain that "guides and controls" the behavior of different cells in the brain. The "no homunculus" argument says that there couldn't exist a distinct and separate physical entity in the brain that governs the behavior of other cells in the brain. In other words, as the argument goes, there are no "ghosts" in the brain. So any notion of "extracellular control" of synaptic modification (connection weight changes) is not acceptable to this framework. Many scientists support this notion (of cells being autonomous learners) with examples of physical processes that occur without any external "control" of the processes, such as a hurricane.

So, under the connectionist theory of learning, the connection weight $w_{ij}(t)$, after observing the t^{th} training example, is given by: $w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t)$, where $\Delta w_{ij}(t)$ is the weight adjustment after the t^{th} example and is determined by the local learning law. Donald Hebb [6] was the first to propose a learning law for this field of science and much of the current research on neural networks is on developing new learning laws. There are now hundreds of local learning laws, but the most well-known among them are back-propagation [14,15], ART [5] and RBF networks [9]. To give an example, the back propagation learning law is as follows: $\Delta w_{ij}(t) = - \eta(\partial E/\partial w_{ij}(t)) + \alpha \Delta w_{ij}(t-1)$. Here η is the learning rate (step size) for the weight update at step *t* and α is a momentum gain term. E is the mean-square error of the whole network based on some desired outputs, in a supervised mode of learning, where a teacher is present to indicate what the correct output should be for any given input. Back-propagation is a steepest descent algorithm and $-\partial E/\partial w_{ij}(t)$ is the steepest descent direction (negative of the gradient).

The distinction between memory and learning

Two of the main functions of the brain are memory and learning. There are of course many categories of memory (short term, medium term, long term, working memory, episodic memory and so on) and of learning (supervised, unsupervised, inductive, reinforcement and so on). In order to characterize the learning behavior of the brain, it is necessary to distinguish between these two functions. Learning generally implies learning of rules from examples. Memory, on the other hand, implies simple storing of facts and information for later recall (e.g. an image, a scene, a song, an instruction). Sometimes these terms are used interchangeably in the literature, and in everyday life: memory is often confused with learning. But the processes of memorization are different from that of learning. So memory and learning are not the same.

Learning or generalization from examples

Learning of rules from examples involves generalization. Generalization implies the ability to derive a succinct description of a phenomenon, using a simple set of rules or statements, from a set of observations of the phenomenon. So, in this sense, the simpler the derived description of the phenomenon, the better is the generalization. For example, Einstein's $E = MC^2$ is a superbly succinct generalization of a natural phenomenon. And this is the essence of learning from examples. So any brain-like learning algorithm must

exhibit this property of the brain - the ability to generalize. That is, it must demonstrate that it makes an explicit attempt to generalize and learn. In order to generalize, the learning system must have the ability to design the appropriate network.

The problems of connectionism - some major misconceptions about the brain

A misconception - no synaptic change signals are allowed to the cells from other sources within the brain

The notion that each neuron or cell in the brain is an "autonomous/independent learner" is one of the fundamental notions in this field. Under this notion, it is construed that individual cells modify their synaptic strengths (connection weights) based solely on their "input and output behavior." The input and output information of a cell may include information about the error in the performance of the given task by the network and an individual cell's contribution to that error; see for example the back-propagation learning law in the last section. This notion implies that no other physical entity external to the cell is allowed to "signal" it to adjust its connection strengths. All of the well-known local learning laws developed to date most faithfully adhere to this notion [2, 3, 5, 6, 7, 9, 10, 14, 15]. However, there is no neurobiological evidence to support this premise. In fact, there is a growing body of evidence that says that extrasynaptic neuromodulators influence synaptic adjustments "directly" [7]. The neurobiological evidence shows that there are many different kinds of neurotransmitters and receptors and many different cellular pathways for them to affect cellular changes. Cellular mechanisms within the cell are used to convert these "extracellular" signals into long-lasting changes in cellular properties. So the connectionist conjecture that no other physical entity directly signals changes to a cell's behavior is a major misconception about the brain. Beyond the neurobiological evidence, this conjecture is also logically inconsistent, as discussed later.

Another misconception - the brain does not collect and store any information about the problem prior to actual learning

In connectionism, brain-like learning algorithms cannot store any training examples (or any other information, for that matter) explicitly in its memory (in some kind of working memory, that is, that can be readily accessed by the learning system in order to learn) [2, 3, 5, 6, 7, 9, 10, 14, 15]. The learning mechanism can use any particular training example presented to it to adjust whatever network it is learning in, but must forget that example before examining others. This is the so-called "memoryless learning" property, where no storage of facts/information is allowed. The idea is to obviate the need for large amounts of memory to store a large number of training examples or other facts. Although this process of learning is very memory efficient, it can be very slow and time-consuming, requiring lots of training examples, as demonstrated in [11,12]. However, the main problem with this notion of memoryless learning is that it is completely inconsistent with the way humans actually learn; it violates very basic behavioral facts. Remembering relevant facts and examples is very much a part of the human learning process; it facilitates mental examination of facts and information that is the basis for all human learning. And in order to examine facts and information and learn from it, humans need memory; they need to remember facts. But connectionism has no provision for it.

There are other logical problems with the idea of memoryless learning. First, one cannot learn (generalize, that is) unless one knows what is there to learn (generalize). And one can find out what is there to learn "only by" collecting and storing some information about the problem at hand. In other words, no system,

biological or otherwise, can "prepare" itself to learn without having some information about what is there to learn (generalize). And in order to generalize well, one has to look at a whole body of information relevant to the problem, not just bits and pieces of information at a time as is allowed in memoryless learning. So the notion of "memoryless learning" is a very serious misconception in these fields, and is totally inconsistent with external observations of the human learning process.

A third misconception - the brain learns instantly from each and every learning example presented to it

A major dilemma for this field is explaining the fact that sometimes human learning is not instantaneous, but may occur much later, perhaps at a distant point in time, based on information already collected and stored in the brain. The problem lies with the fundamental belief in the connectionist school that the brain learns "instantaneously." Instantaneous, that is, in the sense that it learns promptly from each and every learning example presented to it by adjusting the relevant synaptic strengths or connection weights in the network. And it even learns from the very first example presented to it! The learning, as usual, is accomplished by individual neurons using some kind of a "local learning law." Note that "instantaneous learning" is simply a reflection of "memoryless learning;" just the opposite side of the same coin.

A fourth misconception - the networks are predesigned and externally supplied to the brain; and the learning parameters are externally supplied too

Another major dilemma for this field is explaining the fact that a network design, and other types of algorithmic information, has to be externally supplied to some of their learning systems, whereas no such information is externally supplied to the human brain. In fact, not just one, but many different network designs (and other parameter information) are often supplied to these learning systems on a trial and error basis in order for them to learn [5, 9, 10, 14, 15]. However, as far as is known, no one has been able to supply any network design or learning parameter information to a human brain. Plus, the whole idea of "instantaneous and memoryless learning" is completely inconsistent with their trial and error learning processes; there is supposed to be no storage of learning examples in these systems for such a trial and error process to take place. In other words, no such trial and error process can take place unless there is memory in the system, which they disallow.

In order for humans to generalize well in a learning situation, the brain has to be able to design different networks for different problems - different number of layers, number of neurons per layer, connection weights and so on - and adjust its own learning parameters. The networks required for different problems are different, it is not a "same size fits all" type situation. So the networks cannot come "pre-designed" in the brain; they cannot be inherited for every possible "unknown" learning problem faced by the brain on a regular basis. Since no information about the design of the network is ever supplied to the brain, it implies that network design is performed internally by the brain. Thus it is expected that any brain-like learning system must also demonstrate the same ability to design networks and adjust its own learning parameters without any outside assistance. But the so-called autonomous learning systems of connectionism depend on external sources to provide the network design to them; hence they are inherently incapable of generalizing without external assistance. This implies again that connectionist learning is not brain-like at all.

Other logical problems with connectionist learning

But there are more problems with these connectionist ideas. Strict autonomous local

learning implies pre-definition of a network "by the learning system" without having seen a single training example and without having any knowledge at all of the complexity of the problem. There is no system, biological or otherwise, that can do that in a meaningful way; it is not a "feasible idea" for any system. The other fallacy of the autonomous local learning idea is that it acknowledges the existence of a "master system" that provides the network design and adjusts the learning parameters so that autonomous learners can learn. So connectionism's autonomous learners, in the end, are directed and controlled by other sources after all! So these connectionist ideas (instantaneous learning, memoryless learning and autonomous local learning) are completely illogical, misconceived and incompatible with what can be externally observed of the human learning process.

Conclusions

One of the "large" missing pieces in the existing theories of artificial neural networks and connectionism is the characterization of an autonomous learning system such as the brain. Although Rumelhart [15] and others have clearly defined (conjectured) the "internal mechanisms" of the brain, no one has characterized in a similar manner the external behavioral characteristics that they are supposed to produce. As a result, the field pursued algorithm development largely from an "internal mechanisms" point of view (local, autonomous learning, memoryless learning, and instantaneous learning) rather than from the point of view of "external behavioral characteristics" of human learning. That flaw is partly responsible for its current troubles. It is essential that the development of learning algorithms be guided primarily by the need to reproduce a set of sensible, well-accepted external characteristics. If that set of external characteristics cannot be reproduced by a certain conjecture about the internal mechanisms, than that conjecture is not valid.

This article essentially described some of the prevailing notions of connectionism and showed their logical inconsistencies and how they fail to properly account for some very basic aspects of human learning. So there is definitely a need for some new ideas about the internal mechanisms of the brain. From the last two debates and from recent neurobiological evidence, it appears that a very convincing argument can be made that there are subsystems within the brain that control other subsystems. This "control theoretic" notion, which allows external sources to directly control a cell's behavior and perform other tasks, is finding growing acceptance among scientists [13]. This notion has many different labels at this point: non-local means of learning, global learning and so on. It would not be fair if it is not acknowledged that such control theoretic notions are already used, in one form or another, in almost all connectionist learning systems. For example, all constructive learning algorithms (e.g. [5, 9, 10]) use non-local means to "decide" when to expand the size of the network. And the back-propagation algorithm itself [14, 15] depends on a non-local, external source to provide it the design of a network in which to learn. So connectionist systems inadvertently acknowledge this "control theoretic" idea, by using a "master or controlling subsystem" that designs networks and sets learning parameters for them. In other words, as baffling as it may sound, the control theoretic ideas have been in use all along; they are nothing new. Only recently has such non-local means of learning been used effectively to develop robust and powerful learning algorithms that can design and train networks in polynomial time complexity [1, 4, 11, 12]. Polynomial time complexity, by the way, is an essential computational property for brain-like autonomous learning systems.

In addition, a control theoretic framework

resolves many of the problems and dilemmas of connectionism. Under such a framework, learning need no longer be instantaneous and can wait until some information is collected about the problem. Learning can always be invoked by a controlling subsystem at a later point in time. This would also facilitate understanding the complexity of the problem from the information that has been collected and stored already. Such a framework would also resolve the network design dilemma and the problems of algorithmic efficiency that have plagued this field for so long [1, 4, 11, 12]. So one can argue very strongly for such a theory of the brain both from a computational point of view and from the point of view of being consistent with externally observed human learning behavior.

Acknowledgements: I thank Fred Glover, Leon Lasdon, and others for many suggestions that improved this article. Fred Glover, in particular, was a significant inspiration behind our work that uses linear programming models to design and train neural networks. In a memo to a well-known cognitive scientist at his university, he called this debate “an epic debate.”

References

1. Bennett, K.P. and Mangasarian, O.L. (1992). Neural Network Training via Linear Programmig. In P.M. Pardalos (ed), *Advances in Optimization and Parallel Computing*, North Holland, Amsterdam.

2. Churchland, P. (1989). On the Nature of Theories: A Neurocomputational Perspective. Reprinted as chapter 10 in Haugeland, J. (ed), Mind Design II, 1997, MIT Press, 251-292.

3. Churchland, P. and Sejnowski, T. (1992). The Computational Brain. MIT Press, Cambridge, MA.

4. Glover, F. (1990). Improved Linear Programming Models for Discriminant Analysis. *Decision Sciences*, 21, 4, 771-785.

5. Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. Neural Networks, 1, 17-61.

6. Hebb, D. O. (1949). The Organization of Behavior, a Neuropsychological Theory. New York: John Wiley.

7. Levine, D. S. (1998). Introduction to Neural and Cognitive Modeling, Hillsdale, NJ: Lawrence Erlbaum.

8. Minsky, M. and Papert, S. (1988). *Perceptrons*. The MIT Press, Cambridge, MA.

9. Moody, J. & Darken, C. (1989). Fast Learning in Networks of Locally-Tuned Processing Units, Neural Computation, 1(2), 281-294.

10. Reilly, D.L., Cooper, L.N. and Elbaum, C. (1982). A Neural Model for Category Learning. Biological Cybernetics, 45, 35-41.

11. Roy, A., Govil, S. & Miranda, R. (1995). An Algorithm to Generate Radial Basis Function (RBF)-like Nets for Classification Problems. Neural Networks, Vol. 8, No. 2, pp. 179-202.

12. Roy, A., Govil, S. & Miranda, R. (1997). A Neural Network Learning Theory and a Polynomial Time RBF Algorithm. IEEE Transactions on Neural Networks, Vol. 8, No. 6, pp. 1301-1313.

13. Roy, A. (1998). Panel discussion at ICNN’97 on connectionist learning. INNS/ENNS/JNNS Newsletter, appearing with Vol. 11, No. 2, of Neural Networks.

14. Rumelhart, D.E., and McClelland, J.L.(eds.) (1986). Parallel Distributed Processing: Explorations in Microstructure of Cognition, Vol. 1: Foundations. MIT Press, Cambridge, MA., 318-362.

15. Rumelhart, D.E. (1989). The Architecture of Mind: A Connectionist Approach. Chapter 8 in Haugeland, J. (ed), Mind Design II, 1997, MIT Press, 205-232.

ITORMS
INTERACTIVE TRANSACTIONS of ORMS

ITORMS: Call for Papers

ITORMS is the first electronic journal of INFORMS.

ITORMS uses the interactive electronic medium to deliver timely and updated information employing not just the textual and graphical format but also other visual and algorithmic information. The Interactive Transactions of OR/MS publishes original, scholarly high quality articles and bibliographies that provide a perspective view of the OR/MS discipline and exploit the interactivity offered by the Internet. Interactivity has two dimensions in this context. First, the published articles are constantly updated by the authors (contributing editors) to offer the most recent literature on the topic of interest. Second, interactivity allows the readers to learn not only from the paper being read, but also provide access to other related resources that are available or will be developing on the Internet.

ICS loyal members such as Harvey Greenberg, Chris Jones, and Fred Murphy have published in ITORMS. You are invited to submit a paper for review for publication in ITORMS. Check out the ITORMS web site at http://www.informs.org/Pubs/ITORMS, or contact the Editor, Ramesh Sharda at sharda@okstate.edu

Software Announcement

GeNIe 1.0

We are pleased to announce **GeNIe 1.0**, a free development environment for graphical decision-theoretic models.

At the moment it implements Bayesian networks and influence diagrams with, what we believe to be, a pleasant and reliable user interface. It has hierarchical submodels, a windows-style tree view, Noisy-OR nodes, deterministic nodes, multiple decision nodes, relevance reasoning that includes designating nodes as targets, multiple utility nodes, linearly-additive Multi-Attribute utility nodes (we will have generalized MAU nodes in the future), value of information computation, several Bayesian networks algorithms to choose from, on-screen comments, an extensive beginner-oriented HTML-based help system, and many other useful features that one would want from a development environment for graphical models. GeNIe is the main research and teaching vehicle at Decision Systems Laboratory, so naturally it will evolve as time goes.

GeNIe runs on Windows 95/NT computers. It comes with SMILE (Structural Modeling, Inference, and Learning Engine), a portable library of C++ classes that can be embedded in applications that are based on Bayesian networks and influence diagram models developed using GeNIe. SMILE is compiled so far for Windows 95/98/NT and Unix. If there is sufficient demand, we will consider recompiling it for other environments.

GeNIe: Continued on page 18

Message from the Chair

Harlan P. Crowder
ILOG, Inc
Silicon Valley, USA
crowder@ilog.com

After a lot of tenacious effort, the Computer Sciences Technical Section has a new name; we are now the *INFORMS Computing Society*. The designation of "Society" for INFORMS subgroups is reserved for SIGs and sections that have demonstrated a sustained commitment over time and have significantly advanced a particular aspect of operations research and management sciences. On both counts, the INFORMS Computing Society has earned this special designation.

From the early days of our discipline, the computational aspects of OR/MS have advanced in parallel with computer and computational science. Much of what we take for granted today in computational OR/MS owes a debt to computer science: the invention of sophisticated data structures for efficient algorithms; the application of computational complexity that allows us to understand the capabilities, and limitations, of our problem-solving methods; the adoption of advances in computer programming languages to the expression of mathematical models; and the introduction of computational techniques that mimic natural systems, including simulated annealing from physical sciences and genetic algorithms for the biological sciences.

I have always been especially impressed by how the whole study of computational complexity in the 1970s and early 1980s gave us a fundamental understanding of the relative difficulty of a wide range of OR/MS problems. I once asked Philip Wolfe, OR/MS legend extraordinaire, what he thought was the usefulness of complexity theory on the every-day, real-world application of OR/MS. Phil said that was easy; the practitioner could now confidently march into the boss' office and say, "Boss, I don't have any idea how to solve this problem you asked me to solve. But there are a lot of very smart people that have looked at this same problem, and they don't know how to solve it either!" Phil has a way of getting to the point.

Our new designation as a Society is a special honor, but it also means we have more responsibility. New moves are afoot in INFORMS to give more leadership to subgroups, including the opportunity to have Society meetings on an annual basis. This means that INFORMS Computing Society members will need to work harder and get more involved to make the Society a continuing success. I invite you attend our Society business meetings at INFORMS meetings in Seattle, Cincinnati and Philadelphia, and to get actively involved. And I especially urge you to plan on attending our next big INFORMS Computing Society conference in Cancun, Mexico, the first week of 2000.

GeNIe: continued from page 17

GeNIe and SMILE are free for the taking for research, teaching, and personal use. (We do ask authors of publications in which GeNIe and SMILE played a role to acknowledge it.) They can be downloaded from **http://www2.sis.pitt.edu/~genie.** We believe that the programs will prove useful for the community. May the smiling GeNIe grant you your three wishes :-).

Marek J. Druzdzel (marek@sis.pitt.edu) University of Pittsburgh, Decision Systems Laboratory
School of Information Sciences, Intelligent Systems Program, and Medical Informatics Training Program

New Books

Network Optimization: Continuous and Discrete Models by Dimitri P. Bertsekas, Massachusetts Institute of Technology. Athena Scientific, 1998, 608 pages, ISBN 1-886529-02-7.

This book provides an introductory, yet comprehensive and up-to-date treatment of linear, nonlinear, and discrete network optimization problems, and the analytical and algorithmic methodology for solving them. It also provides a guide to important practical network applications and their algorithmic solution, highlights the interplay between continuous and discrete models, and treats extensively the associated analytical and algorithmic issue at an introductory and accessible level.

Among its special features, the book:

- provides a comprehensive account of the theory and the practical application of the principal algorithms for linear network flow problems, including simplex, dual ascent, and auction algorithms.
- covers extensively the main algorithms for specialized network flow problems, such as shortest path, max-flow, assignment, and traveling salesman.
- develops the main methods for nonlinear network problems, such as convex separable and multicommodity flow problems arising in communication, transportation, and manufacturing contexts.
- describes the main models and algorithmic approaches for integer constrained network problems.
- contains many examples, practical applications, illustrations, and exercises.

For preface and table of contents, see *http://world.std.com/~athenasc/netsbook.html*

Reinforcement Learning: An Introduction by Richard S. Sutton, AT&T Shannon Laboratories and Andrew G. Barto, University of Massachusetts. MIT Press, 1998, 344 pages, ISBN 0-262-19398-1.

Reinforcement learning, one of the most active research areas in artificial intelligence, is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment. This book provides a clear and simple account of the key ideas and algorithms of reinforcement learning. The discussion ranges from the history of the field's intellectual foundations to the most recent developments and applications. The only necessary mathematical background is familiarity with elementary concepts of probability.

The book is divided into three parts. Part I defines the reinforcement learning problem in terms of Markov decision processes. Part II provides basic solution methods: dynamic programming, Monte Carlo methods, and temporal-difference learning. Part III presents a unified view of the solution methods and incorporates artificial neural networks, eligibility races, and planning. The final chapter presents a number of case studies.

Combinatorial Optimization by William J. Cook, Rice University, William H. Cunningham, University of Waterloo, William R. Pulleyblank, IBM Research, and Alexander Schrijver, Centrum voor Wiskunde en Informatica. John Wiley and Sons, Inc, 1998, 368 pages, ISBN 0-471-55894-X.

One of the youngest, most vital areas of applied mathematics, combinatorial optimization integrates techniques from combinatorics, linear programming, and the theory of algorithms. Because of its success in solving difficult problems in areas from telecommunications to VLSI, from product distribution to airline crew scheduling, the field has seen a ground swell of activity in the past decade.

Combinatorial Optimization is an ideal introduction to this mathematical discipline for advanced undergraduates and graduate students of discrete mathematics, computer science, and operations research. Written by a team of recognized experts, the text offers a thorough, highly accessible treatment of both classical concepts and recent results. The topics include network flow problems, optimal matching, integrality of polyhedra, matroids and NP-completeness. The book features logical and consistent exposition, clear explanation of basic and advanced concepts, many real-world examples, and helpful, skill-building exercises.

Advances in Sensitivity Analysis and Parametric Programming edited by Tomas Gal, Fern Universität and Harvey J. Greenberg, University of Colorado at Denver. Kluwer Academic Press, 1997, 610 pages, ISBN 0-7923-9917-X.

Since the beginning of linear programming a half century ago, sensitivity analysis has been an integral part of its theory, implementations and applications. As much as possible, these foundations have, over the years, been extended to nonlinear, integer, stochastic, multi-criteria, and other mathematical programming, though it is considered that those advances have so far not provided as rich a body of knowledge. Recent advances in mathematical programming have opened up new insights about sensitivity analysis (including sensitivity analysis in the presence of degeneracy in linear programming). The paradigm, What if....? question is no longer the only question of interest. Often, we want to know Why....? and Why not....? Such questions were not analyzed in the early years of Mathematical Programming to the same extent that they are now, and we have not only expanded our thinking about "post-optimal analysis", but also about "solution analysis", even if the solution obtained is not optimal. Therefore, it is now time to examine all the recent advances on sensitivity analysis and parametric programming.

This book bridges the origins of sensitivity analysis with the state-of-the-art. It covers much of the traditional approaches with a modern perspective (including degeneracy graphs and the tolerance approach). It shows recent results using the optimal partition approach, stemming from interior point methods, for both linear and quadratic programming. It examines the special case of network models, and presents a neglected topic, qualitative sensitivity analysis. The book also presents advances in sensitivity analysis outside of standard linear programming. These include mixed integer programming, nonlinear programming, multi-criteria mathematical programming, stochastic programming, quadratic programs, and fuzzy mathematical programming.

Journal on Computing: Vol. 10, No. 4

Feature Article
"The World Wide Web: Opportunities for Operations Research and Management Science", Bhargava and Krishnan
Commentaries
"Dynamic, Distributed, Platform Independent OR/MS Applications -- A Network Perspective", Bradley and Buss
"Predictions for Web Technologies in Optimization", Fourer
"A New Horizon for OR/MS", Geoffrion
"The World Wide Web: It's the Customers", Trick
Rejoinder
"OR/MS, Electronic Commerce, and the Virtual INFORMS Community", Bhargava and Krishnan
Contributed Research Articles
"Partitioning the Attribute Set for a ProbabilisticReasoning System", Sarkar and Ghosh
"An Efficient Algorithm for Solving an Air Traffic Management Model of the National Airspace System", Boyd, Burlingame and Lindsay
"Lifted Cover Inequalities for 0-1 Integer Programs: Computation", Gu, Nemhauser and Savelsbergh
"A Branch and Bound Algorithm for the Stability Number of a Sparse Graph", Sewell
"Distribution Estimation using Laplace Transforms", Harris and Marchal

Feature Article Abstract: The World Wide Web has already affected OR/MS work in a significant way, and holds great potential for changing the nature of OR/MS products and the OR/MS software economy. Web technologies are relevant to OR/MS work in two ways. First, the Web is a multimedia communication system. Originally based on an information pull model, it is -- critical for OR/MS -- being extended for information push as well. Second, it is a large distributed computing environment in which OR/MS products -- interactive computational applications -- can be made available, and interacted with, over a global network. Enabling technologies for Web-based execution of OR/MS applications are classified into those involving client-side execution and server-side execution. Methods for combining multiple client-side and server-side technologies are critical to OR/MS's use of these technologies. These methods, and various emerging technologies for developing computational applications give the OR/MS worker a rich armament for building Web-based versions of conventional applications. They also enable a new class of distributed applications working on real-time data. Web technologies are expected to encourage the development of OR/MS products as specialized component applications that can be bundled to solve real-world problems. Effective exploitation, for OR/MS purposes, of these technological innovations will also require initiatives, changes, and greater involvement by OR/MS organizations.

Fred Glover Wins von Neumann Prize

Fred Glover was awarded the INFORMS John von Neumann Theory Prize at the Spring 1998 meeting in Montreal. The presentation was made by Leon S. Lasdon, chair of the INFORMS von Neumann Committee (see photo at left). The award was presented to Dr. Glover for "his fundamental contributions to integer programming, networks, and combinatorial optimization." The full text of the presentation citation can be found at *http://www.informs.org/Prizes/vonneumanndetails.htm#1998*.


Fred Glover (left) and Leon Lasdon

Fred's accomplishments are also nicely outlined in the June 1998 issue of OR/MS Today.

The von Neumann prize is awarded annually to a scholar who has made fundamental, sustained contributions to theory in operations research and the management sciences. It is awarded for a body of work, typically published over a period of several years, reflecting contributions that have stood the test of time. The criteria for the prize are broad, and include significance, innovation, depth, and scientific excellence. The award is $5,000, a medallion and a citation.

Computing and Optimization Tools
for the New Millennium

Seventh INFORMS Computing Society Conference

Cancún, México
January 5-7, 2000

Computer science and operations research share an important part of their history. Their interface is responsible for advances that could have not been achieved in isolation. The first six INFORMS Computing Society (ICS) conferences witnessed fascinating developments in the computer science/operations research interface. We would like to take this opportunity to invite you to the seventh ICS conference, which has the goal of bringing together researchers and practitioners in Operations Research, Computer Science, Management Science, Artificial Intelligence, and other related fields. These researchers and practitioners will be responsible for creating the computing and optimization tools for the new millennium.

The advisory committee for the conference is:
· Bruce Golden (*University of Maryland*)
· Harvey Greenberg (*University of Colorado*)
· Paolo Toth (*University of Bologna*)
· John Hooker (*Carnegie Mellon University*).

The conference hotel will be the Westin Regina Cancún, situated just up the beach from the Punta Nizuc natural reef. Anyone interested in organizing sessions or tutorials should contact either one of us. We hope to continue the tradition of excellence established in previous ICS conferences, for which we need to count with your support and participation. For additional information, please visit the conference web site at http://www-bus.colorado.edu/Faculty/Laguna/cancun2000.html

General Co-Chairs:

Manuel Laguna
University of Colorado
Manuel.Laguna@Colorado.Edu

José Luis González Velarde
Monterrey Tech
lugonzal@campus.mty.itesm.mx

News about Members

Colin Bell (colinbel@microsoft.com) left his position as Professor of Management Sciences and Associate Dean at the University of Iowa for a position in the Microsoft Project Development group at Microsoft Corporation.

Ismail Chabini (chabini@mit.edu) is the winner of an NSF Career award starting in June 1998.

Shyam (schadha@uwec.edu) and **Veena Chadha** at the University of Wisconsin, Eau Claire, with several students, investigated the distribution problem of the Walmart distribution center located at Menomonie, WI in the spring of 1998. A mathematical model was developed using ten cities of the region. This project was funded by the Student Research Collabration Grants of the University of Wisconsin, Eau Claire.

Jonathon Eckstein (jeckstei@rutcor.rutgers.edu) has been granted tenure and promoted to Associate Professor at Rutgers University. In addition, he was awarded the Rutgers University Board of Trustees Fellowship for Scholarly Excellence in May 1998 and his second child was born in March 1998.

Anna Nagurney (nagurney@gbfin.umass.edu) has been named to an endowed chair position at the University of Massachusetts. Her title is John F. Smith Memorial Professor. This professorship is funded by a gift from Jack Smith, CEO of General Motors, to honor the memory of his father.

Marcel F. Neuts (marcel@tucson.sie.arizona.edu) is developing a research program in computer experimentation for probability models. He is also starting a news bulletin in this area - in addition to the existing Stochastic Models News and the Matrix-Analytic Bulletin. For the year 2000, he is planning a special issue of Stochastic Models on the methodology of computer experimentation. He also offers consulting and educational services in applied probability and its computational aspects.

S. Raghavan (suraghav@rhsmith.umd.edu) moved from his position as Acting Director, Optimization Group, at U S WEST Advanced Technologies, Boulder CO, to take up a faculty appointment in the Decision and Information Technologies Department, within the Robert H. Smith School of Business at the University of Maryland, College Park MD.

Upcoming Meetings

INFORMS Spring 1999 Meeting, Cincinnati Convention Center and Omni Netherland Plaza, Cincinnati OH, May 2-5, 1999. Theme: Delivering to the Global Consumer. Features comprehensive coverage of OR/MS topics while also striving for increased emphasis in pedagogy and practice. General Chair David Rogers, (david.rogers@uc.edu), URL=*http://www.informs.org/Conf/Cincinnati99/*

Fourth Conference on Information Systems and Technology, Cincinnati OH, May 2-5, 1999, sponsored by the INFORMS Colleges on Information Systems and Artificial Intelligence. URL=*http://www.cob.ohio-state.edu/~rolland/cist-99/main.htm*

Winter Simulation Conference (WSC '98), Grand Hyatt Hotel, Washington DC, December 13-16, Theme: "Simulation in the 21st Century". URL=*http://www.wintersim.org*

19th IFIP TC7 Conference on System Modelling and Optimization, Cambridge, England, July 12-16, 1999.

Society for Medical Decision Making (SMDM), Reno, NV, October 2-5, 1999.

INFORMS Fall 1999 Meeting, Philadelphia Marriott, Philadelphia PA, November 7-10, 1999.

INFORMS/CSTS 7th Biennial Conference, Cancun Mexico, January 2000. General Chair: Manuel Laguna.

INFORMS Spring 2000 Meeting, San Francisco CA.

INFORMS/KORS, Seoul, South Korea, Summer 2000.

Institute for Operations Research and the Management Sciences

901 Elkridge Landing Road, Suite 400
Linthicum, MD 21090-2909

PRE-SORTED
FIRST-CLASS POSTAGE PAID
BALTIMORE, MD
PERMIT NO. 4537