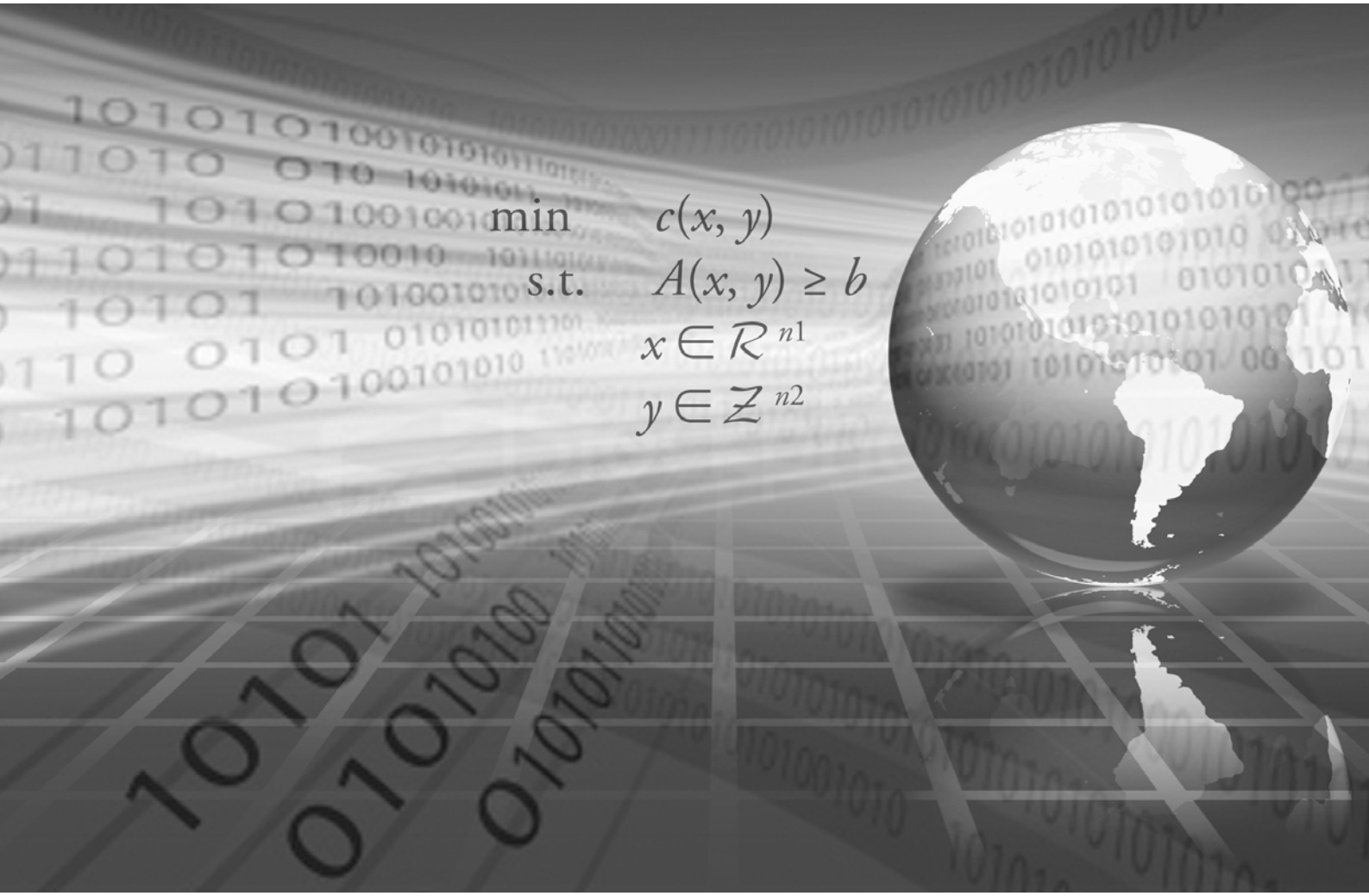
N°77

# OPTIMA

*Mathematical Programming Society Newsletter*

JULY 2008




## Accuracy in computation and benchmarking

This second issue of 2008 has accuracy in computation and benchmarking of computer codes as main themes. The two topics are strictly related. In the scientific contribution François Margot proposes a methodology for testing the accuracy and the effectiveness of cut generators for Mixed-Integer Linear Programming while in the discussion column Matthew Saltzman reports on a very interesting panel discussion on benchmarking held in Puerto Rico during the INFORMS International 2007. We hope that putting these two contributions together in issue 77 of Optima would be beneficial for our readers and the entire Mathematical Programming community since the topics have crucial impact in the area.

Matthew Saltzman's contribution is written "in memoriam" of Lloyd Clarke (1964–2007), a respected member of the Mathematical Programming community and a good friend. Optima 77 is dedicated to him.

Alberto Caprara
Andrea Lodi
Katya Scheinberg

77

# Notes on a Panel Discussion on Benchmarks at INFORMS Puerto Rico, July 2007

**Matthew J. Saltzman**
*Clemson University*
May 22, 2008

The 2007 INFORMS International meeting took place in Puerto Rico in July. One session there was a panel discussion that I organized, entitled *Benchmarking: Who, What, When, Where, Why?* Participants included the late Lloyd Clarke of ILOG, Steve Dirkse of GAMS, Bob Fourer of Northwestern University, Bill Hart of Sandia National Laboratories, Leon Lasdon of the University of Texas, and François Margot of Carnegie Mellon. Carol Tretkoff of ILOG took notes, doing her level best to keep up with the discussion. (Hans Mittleman, an obvious choice for a panelist, was at the EURO conference, which took place that same week in Prague.) I would like to thank all of the participants for taking part in the original discussion and helping me recall what went on. This article is a recap of my impressions of that discussion, and I take responsibility for any inaccuracies, omissions, or distortions of the events.

Participants were free to address any topics they chose. However, I did set up the discussion by posing the title questions. Our main objective was not to resolve issues so much as to raise them, and to consider how we might move forward to start to address them.

**Who, when, why?** *Who should carry out benchmarks and comparisons to ensure that they are reliable, unbiased, and useful? Who makes up the audience for benchmark studies? Under what circumstances should various types of benchmarks or comparisons be carried out? What do researchers, developers, and users expect to learn from benchmarks?*

Benchmarks may be carried out for different audiences. Those carrying them out may have agendas that differ from those using them to decide on algorithms or codes to use when solving a particular problem. For example, developers of new algorithms for a specific problem may compare their codes with existing specialized codes for that problem or general-purpose codes for broader classes of problems.

Potential users of commercial or other general-purpose solvers often have highly specialized workloads, so benchmarks on standard test sets may not provide them with useful selection criteria. Of course, vendors are interested in showing their codes off to best advantage.

One thing is clear: the interests of some people who produce benchmarks can be quite different from the interests of people who consume them. I am not suggesting at all that anything nefarious is going on here. In many cases, it is simply impossible for a vendor or author to address the concerns of all possible customers or readers.

One of Lloyd Clarke's main points regarding benchmarks of commercial codes is that every customer has a unique work profile. A vendor simply cannot provide a single benchmark that addresses all customer workloads, only the customer can provide an appropriate set of problem instances. So customers whose primary interest is in determining which of a competing set of codes should be acquired can only make such a determination by conducting their own tests with representative problem instances from their own collections, on their own machines.

Authors of research articles are naturally interested in showing their codes off in the best possible light. Again, while there need not be any intention to mislead readers, this interest can guide the researcher to produce results consistent with his interests. For example, a researcher might select a suite of test problems, work with his algorithm to get it to perform well on that set of problems, and then use the same problems to carry out the published study. If the study involves a head-to-head comparison with another code, the researcher may not make the same effort to tune the competitor's code. It is quite common, for example, to see a specialized algorithm compared against a general purpose code run with all settings left at their default values.

Authors may not be experts in statistical analysis, so may not perform incisive

analyses. Many computational studies still consist of reporting times on some (arbitrary) collection of instances and reporting solution times on the individual instances and/or the total group. Statistical tests, if they are performed at all, may not be valid because the data may not satisfy the tests' assumptions (e.g., normality).

**What?** *What are components of a fair, unbiased, useful benchmark or comparative study?*

- How should parameter settings be chosen?

  Parameter setting is a challenging problem. General-purpose solvers may have many different parameters and determining good combinations of values for particular problems may require significant testing. Even then, solvers may behave differently from one version to the next, so parameter settings need to be recomputed for new versions. There has been some limited experimentation with dynamic parameter adjustment, but this area of research is very young.

  Even in the case where solvers of similar capability are being compared, the solvers may have non-comparable parameters and tolerances or different default settings for the "same" parameters. François Margot addresses a related problem involving tolerances, accuracy of floating point arithmetic and cutting-plane validity elsewhere in this issue.

- How should head-to-head comparisons be carried out when comparing:
  - general-purpose codes,
  - a special-purpose code against a general-purpose code, and
  - different special-purpose codes?

  Is it fair to compare, say, a special-purpose code with an off-the-shelf solver with all default parameter settings? If not, how might one assess the performance of a special-purpose code?

  What sort of analysis is appropriate when comparing codes? How can a reader understand the results of a comparison?

**Where?** *Where should benchmark results be published? Where can funding be obtained to carry out benchmarking studies?*

Software development and related activities—such as benchmarking—do not conform to the common academic notion of "research", which corresponds to the "scholarship of discovery" (i.e., pursuit of new knowledge for its own sake) in Boyer's taxonomy [1]. Instead, they are more related to "scholarship of integration," "scholarship of application," or "scholarship of teaching and learning." Discovery is the traditional realm of archival journals and federal funding agencies, whereas it is more difficult to gain recognition for integration and the creation of infrastructure. Thus, publishing or funding a benchmark study unaccompanied by creation of a new algorithm or model is problematic.

As a result, generating interest among researchers in carrying out benchmark studies is challenging, even when we agree that such studies are useful. In addition, even if benchmark studies were publishable in archival journals, it is not clear that they would reach the audience that would benefit most.

**Panelist comments.** Panelists tended to focus on two issues: (1) what kinds of measurements and comparisons are useful, and (2) how can large-scale benchmarks be managed efficiently.

François Margot pointed out that it is reasonable to compare run times of two solvers if they generate the same solution. But often, two solvers produce different "optimal" solutions. In that case, we need to compare not just running times, but also solution quality. If the solution one solver generates satisfies looser feasibility or optimality conditions than that of its competitor, then it is not clear that comparing run times is fair. Comparisons are particularly risky for problems that are not well scaled. An audience member pointed out that rescaling problems can often mitigate this difficulty. Margot advocates that solvers provide quality guarantees such as bounds on the solution's distance from feasibility, optimality, integrality.

Leon Lasdon pointed out that there are facilities in GAMS that make benchmarking simpler to manage.

Bill Hart observed that solution times are not the only factors of interest in performance analysis. Solution quality and reliability are also key factors. While open-source codes have many advantages in an environment where custom solvers are necessary, reliability is an area where open-source codes are not always a match for their commercial counterparts. Sandia runs a large suite of unit tests for reliability. Performance testing is less important, as long as performance is within reason.

Bob Fourer described the NEOS Benchmarking Service. A user can submit a problem instance to the service and have it solved on all solvers of interest, on a single machine.

Fourer also described a useful and currently popular comparison method called *performance profiles* [3]. Performance profiles plot a performance metric $\tau$ on the horizontal axis and, for each solver $s$, the fraction $\rho_s(\tau)$ of test problems with $\log_2 r_{ps} \le \tau$, where $r_{ps}$ is the ratio of solver $s$'s metric on problem $p$ to the best solver's metric on that problem. Thus, $\rho_s(0)$ is the fraction of problems on which solver s performed best, and $\rho_s(\infty)$ is the fraction of problems that solver $s$ solved successfully.

Performance profiles have some appealing features as benchmarking statistics:

- They are not sensitive to the outcomes for a small subset of problems.
- They are not sensitive to small changes in outcome values.
- They provide an indication of the size of the difference in performance.
- They can be used with a variety of metrics.
- They can be used to compare more than two solvers.

As far as I know, however, there is no statistical analysis associated with performance profiles. The profile displays can be compelling, but I am not aware of any quantitative summary of the results or hypothesis tests that can support the impressions given by the graphs. Examples

of a variety of statistical analyses of computational experiments can be found in [2].

Steve Dirkse described the PAVER (Performance Analysis and Visualization for Efficient Reproducibility) performance analysis service, available at the GAMS World Web site (www.gamsworld.org/performance/paver). PAVER accepts GAMS trace files and produces a variety of analyses. Dirkse emphasized reproducibility as a goal for benchmarking and the need for tools and test sets to make benchmarking as painless as possible. Lloyd Clarke reiterated Hart's points regarding benchmarking and quality assurance (which includes regression tests) from a developer's perspective. He emphasized that customers should benchmark their own work profiles, rather than relying on published benchmarks.

**Conclusions.** Because of the variety of needs, there is no easy recipe for benchmarks. Users will need to benchmark using their own workloads, authors will need to publish experiments, developers will need to include regression testing. Several things can be done to improve the quality of results.

- Test harnesses and data sets can be improved to make carrying out experiments easier. PAVER is a step in this direction. Hans Mittleman's benchmark site (http://plato.asu.edu/bench.html) is a useful resource for test problems and other information. There are several other collections of test instances for various problems, but they are often not well known outside of the communities of researchers working on those problems.
- Authors and readers need to be educated on how to carry out experiments and analyse the results. Journals need to set standards and referees need to enforce them in publications. Journals also need to provide facilities to support reproducibility of published results. A new journal—*Mathematical Programming Computation*, edited by Bill Cook of Georgia Tech and to be published by Springer in 2009—will require that referees have access to software and data sets be included with submissions so that results of computational tests can be verified, and will encourage the publication of software as open source.
- The hardest problem is that the academic culture needs to change to give credit for these kinds of activities. Improved benchmarking tools—a part of broader infrastructure development—enhance the progress of research and the incorporation of new results in applied settings.

**In memoriam:** Lloyd Clarke, 1964-2007.

## References

[1] E. Boyer. *Scholarship Reconsidered: Priorities of the Professoriate.* The Carnegie Foundation for the Advancement of Teaching, Stanford, California, 1990.

[2] M. Coffin and M. J. Saltzman, "Statistical Analysis of Computational Tests of Algorithms and Heuristics," *INFORMS Journal on Computing* 12(1), 2000, 24-44.

[3] E. D. Dolan and J. J. Moré, "Benchmarking Optimization Software with Performance Profiles" *Mathematical Programming* 91, 2002, 201-213.

# MIP 2008: Workshop on Mixed Integer Programming

The fifth Workshop on Mixed Integer Programming (MIP 2008) will be held at the Columbia University, New York, during August 4-8, 2008. The workshop is open for participation. The program will be composed of a limited number of invited talks organized in a single track. In addition, there will be a contributed poster session and all participants are invited to submit an abstract for the poster session.

**Speakers**

* Warren Adams — Clemson University
* Kent Andersen — University of Copenhagen
* Alper Atamturk — University of California, Berkeley
* Pasquale Avella — Universita del Sannio
* Christoph Buchheim — University of Cologne
* Emilie Danna — ILOG
* Daniel Espinoza — Universidad de Chile
* Yongpei Guan — University of Oklahoma
* Oktay Günlük — IBM Research
* Ignacio Grossmann — Carnegie Mellon University
* John Hooker — Carnegie Mellon University
* Ellis Johnson — Georgia Institute of Technology
* Bala Krishnamoorthy — Washington State University
* Adam Letchford — Lancaster University
* Sven Leyffer — Argonne National Laboratory
* Peter Malkin — University of California-Davis
* Ronald Rardin — University of Arkansas
* Gabriel Tavares — Dash Optimization
* Rekha Thomas — University of Washington
* Mike Trick — Carnegie Mellon University
* Hamish Waterer — University of Auckland, New Zealand
* Ruriko Yoshida — University of Kentucky

**Program Committee**

Simge Kucukyavuz, Quentin Louveaux, Andrew Miller, Gabor Pataki, Jean-Philippe Richard

**Local Committee**

Daniel Bienstock, Oktay Günlük.

Conference Web page: http://coral.ie.lehigh.edu/mip-2008/index.html

We encourage you to secure lodging as soon as possible — prices

# Who will host ISMP 2012?

## Call for site proposals

The triennial International Symposium on Mathematical Programming (ISMP) is the flagship event of our Society. It regularly gathers over a thousand confirmed and young scientists from all over the word, representing all areas, theory and applications of the field. Hosting such an event thus represents a big challenge and a vital service to the community. Of course it also has a lasting effect on the visibility of the hosting institution. This call for proposals is addressed at local groups willing to take up that challenge.

Preliminary bids will be examined by the Symposium Advisory Committee (SAC) , which will then issue invitations for detailed bids. The final decision will be made and announced during the upcoming ISMP in Chicago. The SAC is composed of Abilio Lucena, Kazuo Murota, Rüdiger Schultz, David Williamson, Laurence Wolsey, and chaired by Thomas Liebling. Preliminary bids should be brief and contain some info about

1. The site
2- Facilities (traditionally the symposium venue has been a university campus)
3- Logistics, travel, accommodation, transportation,...
4- Likely local organizers

Please address your preliminary bids until September 15, 2008 to the SAC chair thomas.liebling@epfl.ch, who will also provide further indications at request.

# Testing Cut Generators for MILP

**François Margot**
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213
fmargot@andrew.cmu.edu
March 2008

## Abstract

We present a methodology for testing the accuracy and strength of cut generators for mixed-integer linear programming. The proposed procedure first assesses the accuracy of the generator and then compares the strength of generators with similar accuracy.

## 1 Introduction

Empirical testing of cut generators for Mixed-Integer Linear Programming (MILP) is a problem appearing in various settings. In its simplest form, it occurs when studying a new class of cutting planes, or when a new implementation of a cut generator needs to be compared with an existing one. A related situation is the problem of testing different cutting strategies, where decision about which cut generators to apply and how to set their respective parameters. Note also that generators need to be compared on speed (Does the generator help solving a problem faster?) and accuracy (How often does the generator generate invalid cuts?). The latter point, in particular, seems to have been completely ignored in the literature. Developers usually add various safeguards to avoid as much as possible the generation of invalid cuts due to limited numerical accuracy, but the effectiveness and pertinence of these safeguards is rarely discussed and almost never supported by data. Papers studying provably valid cut generation in finite precision arithmetic have appeared recently [6, 20] and development of codes working in infinite precision arithmetic is also underway [1, 7, 12].

Here, we assume that a given (bug free) cut generator is given and we want to obtain an empirical measure on how often it generates invalid cuts due to the limitations of finite precision arithmetic computations. We also describe a methodology to compare the strength of several cut generators going beyond the customary reporting of "average gap closed at the root" or average run time of a branch-and-cut code using these generators on sample problems.

There are obvious limitations on the information obtained from either of these approaches: For the first approach, the strength of the cuts at the root node is not always a good indicator of the usefulness of a family of cuts, there is a trade off between the quality of the lower bound and the time devoted to obtain it, and it is mostly useless to test the accuracy of the cut generator. For the second approach, most papers are happy when the code finds the optimal solution of the problem, assuming implicitly that everything went fine. However, it might well be the case that the optimal solution is found early in the search by a heuristic algorithm (or that several optimal or near-optimal solutions exist and one of them survived). What happens afterwards with the cut generators is then moot. To push things to the extreme, having a cut generator that generates invalid cuts can even appear positive, as the solution time might go down significantly. To avoid this problem, turning off the heuristic algorithm is a possibility, with the drawback that a problem that could be solved in minutes requires a much longer solution time. Another problem is the interaction with other cut generators. It is well-known for example that Mixed-Integer Gomory cuts [8], Mixed-Integer Rounding cuts [19], and Disjunctive cuts [2, 11] are equivalent [13]. It is then difficult to assess the contribution of one of these cut generators when the others can pick up the slack. On the other hand, trying to solve problems with a branch-and-cut using a single cut generator and no heuristic algorithm is likely to require unacceptably large computation time.

A major reason why the second approach dominates the empirical papers is that it benchmarks the "true" objective: solving problems as fast as possible. This is certainly appropriate when benchmarking branch-and-cut codes. However, as pointed out by Hooker in [9, 10], this type of empirical results yield very little insight on how to improve the tested algorithms. Devising specific experiments to illustrate properties or weaknesses of the algorithms is the major point of a true empirical analysis of algorithms. McGeoch develop this idea in several papers [15, 16, 17, 18].

The goal of this paper is to describe a methodology for testing cut generators that avoids most of the pitfalls of the

usual approaches and is in the spirit of the empirical analysis of algorithms of Hooker. It is based on solving sample MILP problems repeatedly, using a random branching rule and diving towards a known feasible solution. The introduction of a random component allows for the generation of many data points from a single instance and for meaningful statistical testing of the results. This paper intends to convey the main ideas of the testing procedure and additional details can be found in the technical report [14].

The paper is organized as follows. Section 2 describes the proposed method for testing a cut generator. This method, random diving towards a feasible solution, requires the knowledge of a feasible solution of the problem, where the feasibility requirement is much stricter than the commonly used "not violating too much any constraint". This requirement is quite difficult to meet for many usual benchmark MILP problems (see [14] for details). Section 3 then reports results obtained for seven Gomory cut generators. First, these generators are compared for accuracy in Section 3.1 and three of them are deemed having similar accuracy. These three generators are then compared for strength in Section 3.2. Conclusions are given in Section 4.

## 2 Random Diving Towards 0-Feasible Solutions

Consider the instance $I$ of an MILP problem:

$$\begin{array}{ll} \min & c(x, y) \\ \text{s.t.} & A(x, y) \geq b \\ & x \in \mathcal{R}^{n1} \\ & y \in \mathcal{Z}^{n2} \end{array} \qquad (1)$$

Let $(x^I, y^I)$ be a solution of $I$ and let $\varepsilon \geq 0$. The solution is $\varepsilon$-*integer* if each entry in $y^I$ is within $\varepsilon$ of an integer value. The solution is $\varepsilon$-*feasible* if it is $\varepsilon$-integer and the absolute violation of any of the constraints $A(x^I, y^I) \leq b$ is at most $\varepsilon$.

It is quite difficult to find how often a cut generator for MILP generates invalid cuts. We suggest to estimate this by generating a set $S$ of feasible integer solutions and testing how often one of them is cut. In order to have a valid test, it is necessary that the solutions in $S$ are 0-feasible solutions, as any solution that if not $\varepsilon$-feasible for some $\varepsilon > 0$ might correctly be cut by a generated cut. This is more than a minor problem, as MILP solvers return slightly infeasible solutions on most problems. The only way we found to generate 0-feasible solutions for MILP problems is an iterative procedure that find an almost 0-feasible solution and modifies the right hand side of violated constraints until a 0-feasible solution for the modified problem is obtained.

In this section, just assume that we have a 0-feasible solution $(x^I, y^I)$ of an instance $I$ of (1). Assume that we want to test the accuracy of a cut generator. We can dive towards the solution, while using the cut generator, and record if the solution is still $\varepsilon$-feasible or not for some value of $\varepsilon$ (we use $\varepsilon = 10^{-6}$ in the tests). More precisely:

1. Start with the LP relaxation of $I$; flag := 0.
2. Repeat
   2.1 Repeat $k$ times
      2.1.1. Generate and apply cuts.
      2.1.2. Resolve the LP.
      2.1.3. If the LP is infeasible then flag := 2 and stop.
      2.1.4. Otherwise, let $(\bar{x}, \bar{y})$ be the optimal LP solution.
   2.2. If $(x^I, y^I)$ is not $\varepsilon$-feasible then flag := 1.
   2.3. If $(\bar{x}, \bar{y})$ is $\varepsilon$-integer then stop.
   2.4. Otherwise select randomly an index $j$ with $\bar{y}_j$ fractional.
   2.5. Set $y_j := y^I_j$ in the LP.
   2.6. If a time limit is reached then flag := 3 and stop.

Algorithm 1: Diving towards a 0-feasible solution.

The algorithm either terminates with `flag = 0`, meaning that the LP relaxation has an $\varepsilon$-integer feasible optimal solution and $(x^I, y^I)$ is still $\varepsilon$-feasible, or it raises one of three types of failure indicated by the value of `flag`:

- `flag = 1`: $(x^I, y^I)$ is no longer $\varepsilon$-feasible, but another $\varepsilon$-integer feasible solution is reached.
- `flag = 2`: The LP relaxation is infeasible.
- `flag = 3`: The time limit is reached.

Terminating with `flag = 1` or `flag = 2` is annoying, but reaching `flag = 1` is sometimes less severe than reaching `flag = 2`, as a slight alteration of the values of the continuous variables $x^I$ might restore $\varepsilon$-feasibility.

Notice that it is possible that the algorithm terminates in step 2.3 with a solution $(x, \bar{y})$ that is $\varepsilon$-integer but not $\varepsilon$-feasible. This indicates a lack of precision in the LP solver, something that is unrelated to the cut generator. It seems thus fair not to penalize the cut generator by reporting a failure in this case. On the other hand, it could happen that the algorithm stops is step 2.1.3, reporting incorrectly that the LP is infeasible due to a similar lack of precision in the LP solver. If $(x^I, y^I)$ is then still $\varepsilon$-feasible, it might be unfair to report a failure for the cut generator. However, this case seems quite unlikely to happen. Note that these problems could be avoided by using an exact LP solver such as `QSopt ex` [1], `Lpex` [7], or `perPlex` [12]. However, if the cut generator is intended to be used with a nonexact LP solver, it is unclear which experiment setting gives the most pertinent information. Experiments in this paper are obtained using a non-exact LP solver, `Clp` [5] and the intent is also to test how the generated cuts might create problems for the LP solver.

The above scheme has several interesting features: First, the randomization in step 2.4 allows for statistical testing. From one instance with a few hundred of integer variables, one can generate many observations. It is also possible to make statistics on cut generation time, LP resolve time, evolution of the lower bound, and, of course, the failure types. In addition, the number of variables set to integer values in step 2.4 before reaching an integer solution can be used as a measure of the strength of the cuts: The purpose of using cuts is to reduce the size of the enumeration tree. Stronger cuts should yield shorter paths from the root to the leaves of the tree, and this should be reflected in the average length of paths observed while diving towards a feasible solution. This test is devised to test the accuracy of the cuts, not to predict the power of a cut generation strategy in a branch-and-cut algorithm.

## 3 Application Example

As an example of application of Algorithm 1, tests on variants of two cut generators are reported (the paper [14] compares four generators). These generators both implement Gomory mixed-integer cut generators:

1. CglGomory: The default Gomory cut generator of the Cut Generation Library (Cgl) of COIN-OR [5]. This generator is denoted by G in the remainder.
2. CglGomory2: A Gomory cut generator written by the author that has many parameters. This generator uses optimal tableau information provided by the LP solver whereas G recomputes the optimal tableau from the optimal basis information. This generator is denoted by G2 in the remainder.

As mentioned in the previous section, applying this formula blindly to generate cuts is likely to generate some invalid cuts. This is why the generators listed above have ways to prevent the generation of invalid cuts as well as for discarding small coefficients in a cut. Generator G2 is fully parametrized making easy to use it with different values of the parameters. Define the *dynamism* of a cut is the ratio between the smallest and largest absolute values of its nonzero coefficients. The generators have a threshold LUB for deciding that a variable upper bound is large. The parameters that are modified in the experiments and their default values are:

- MAXDYN = 1e8: a cut is discarded if none of the variables with nonzero coefficient have a large upper bound and its dynamism is larger that this value.
- MAXDYN LUB = 1e13 similar to MAXDYN, but for cuts where some of the variables with nonzero coefficients have a large upper bound.
- AWAY = 0.05: let $f_0$ be the fractional part of the basic variable associated with a row of the optimal tableau; the row is not used to generate a cut if $f_0$ is not at least AWAY from an integer value.
- MINVIOL = 1e-7: If the violation of the cut by the current optimal LP solution is lower than this number, the cut is discarded.

In addition to the default settings above, five variants of G2 are tested. All variants are more restrictive than the default generator and should generate less and "safer" cuts. Variants are labeled G2P1 through G2P5 and with parameters set as in the default setting except the following: P1 has MINVIOL = 1e-4, P2 has MINVIOL = 1e-2, P3 has AWAY = 0.08, P4 has MAXDYN = 1e4 and MAXDYN LUB = 1e8, and P5 has MAXDYN = 1e6 and MAXDYN LUB = 1e10.

All generators are set so that there is no limit on the number of cuts they generate and no limit on the number of nonzero coefficients in generated cuts. All results are obtained using $k = 10$ in Algorithm 1.

Before discussing the results, let us make it clear that the settings used above (in particular having no limit on the number of nonzero coefficients in a cut) are chosen to put stress on the generators and LP solver. Using 10 rounds of cutting after each fixing of a variable is also probably not the optimal setting for using these generators in a branch-and-cut code. Nevertheless, the comparison across the fourteen variants considered is a fair one.

## 3.1 Comparing Accuracy

The goal of this section is to compare the accuracy of the seven generators on the MIPLIB3_C instances. These sixteen instances are modifications of some of the benchmark problems from MIPLIB3 [3]. We use Algorithm 1 with $k = 10$ and 20 trials for each 0-feasible solution for a total of 5,500 trials for each generator.

Table 1 reports the value of flag at the end of Algorithm 1. In term of success (i.e. flag = 0), the winner is G2P4. For failures of types 1 or 2 (the most critical ones) G perform best, but the number of trials that it is unable to complete within the time limit (10 minutes cpu) is much larger. There is an obvious difference in the failure patterns for these algorithms. A partition of them into the four groups {G}, {G2, G2P1}, {G2P2, G2P3}, and {G2P4, G2P5} seems a fair grouping. In term of failure 1 and 2, groups {G}, and {G2P4, G2P5} are similar, but the latter solve many more instances within the time limit.

| | Flag | | | |
|---|---|---|---|---|
| name | 0 | 1 | 2 | 3 |
| G | 5332 | 1 | 6 | 161 |
| G2 | 5315 | 0 | 39 | 146 |
| G2P1 | 5359 | 0 | 38 | 103 |
| G2P2 | 5390 | 0 | 26 | 84 |
| G2P3 | 5440 | 0 | 23 | 37 |
| G2P4 | 5473 | 0 | 11 | 16 |
| G2P5 | 5456 | 0 | 10 | 34 |

Table 1: Gomory cut generators comparison on MIPLIB3_C instances.

Interesting insights on the behavior of the generators can be obtained by studying the repartition of trials ending with flag > 0 among the different instances. Some instances (such as gt2_c) create problems for all cut generators, each of them failing at least twice. This instance has 188 variables, 24 of them binary, 29 constraints with a maximum absolute value for right-hand side of about 6,600, all variables bounds at most 15 in absolute value, dynamism of 152, and a maximum absolute value of 9 for the entries of the 0-feasible solutions used.

## 3.2 Comparing Strength

While Algorithm 1 is designed to test the accuracy of a generator, it is possible to get information about the strength of the cuts by performing statistical tests on the number of variables set to integer values in step 2.4 in each trial. Comparing performances of algorithms by statistical tests is well covered in the literature. We refer the reader to [4] for an excellent introduction to the topic. The basic test commonly used when comparing performances is a t-test. However, when comparing more than a pair of algorithms, other tests have been devised. In this paper, we use Tukey's Honest Significant Differences test (*THSD test*). Both the t-test and THSD t-test are based on Analysis of Variance (*ANOVA*). The statistical design used for our application is a two-way factorial design with three factors: "algorithm", "instance", and "solution". The factor "solution" is embedded in the factor "instance", and the factor "algorithm" is crossed with "instance/solution". For each value of "algorithm", "instance" and "solution", we have 20 observations for the number of variables fixed to integer values. The observations for runs that fail are removed. Hence, if none of the observation results in a failure, we have a balanced design. Otherwise, the design is slightly unbalanced, assuming that only a low percentage of runs end with a failure. This is supported by the tables listed in Section 3.1. Both ANOVA and THSD might give misleading results with unbalanced designs, but can handle slightly unbalanced designs. Moreover, we are mostly interested in the effect associated with the factor "algorithms", and the ANOVA computations can be trusted for the main factor, even with unbalanced designs. The results below are obtained

using the statistical package R [21] version 2.5.1 (2007-06-27). The ANOVA results show that the effect associated with the factor "algorithm" is significant with 95% confidence.

The results of the THSD test are given in Table 2. A "+" (resp. "–") entry in row *A* and column *B* means that algorithm A required more (resp. less) variables to be fixed than algorithm *B* with a significance threshold of 95%. A (".") entry means that no conclusion can be drawn from the results. A total order of the algorithms can be derived from Table 2: G is superior to G2P4 which is superior to G2P5.

| | G | G2P4 | G2P5 |
|---|---|---|---|
| G | · | − | − |
| G2P4 | + | + | − |
| G2P5 | + | + | · |

Table 2: THSD results for generators G, G2P4, and G2P5, on the MIPLIB3 C instances.

## 4 Conclusions

Comparing cut generators for MILP is not an easy matter. One might want to compare speed of generation, speed of reoptimization after adding a round of cuts, or strength of the generated cuts. Testing for strength, in particular, is difficult. The contention of this paper is that comparing strength of cut generators without a sense of how accurate the generators are is not very informative. While one could try to devise a method to test simultaneously accuracy and strength, we propose here to first assess the accuracy of a cut generator and then compare strength of cut generators that have similar accuracy.

The proposed method, random diving towards a feasible solution, has the attractive feature that its results depend only on the cut generator and the precision of the LP solver. While the latter dependency is unfortunate (but could possibly be removed by using an exact LP solver), the dependency on algorithmic parts outside the cut generator is far smaller than in any other test that we are aware of. As is usual when testing numerical precision of algorithms, the results might also depend on the machine and compiler used in the tests. The contribution of this paper is thus more the testing method than the ranking of the generators obtained in Section 3. The dependency of the ranking obtained on the choice of the sample problems unfortunately prevents to draw conclusions on the relative strength of families of cuts in general. This weakness of the proposed method does not seem easy to remove.

Another interesting feature of the method is that analyzing the results raises many interesting questions directly related to improving the performances of a cut generator. For example, studying why failures occur on an apparently innocuous instance such as gt2_c might suggest new way to prevent the generation of invalid cuts. Investigating how aggressive one can be with the parameter setting, yet keeping a low probability of generating invalid cuts, is a question with important practical implications, in particular if this can be linked to properties of the instance. The method is well-suited to explore such questions.

## References


[1] Applegate D.L., Cook W., Dash S., Espinoza D.G., "Exact Solutions to Linear Programming Problems", preprint, (2006).

[2] Balas E., "Disjunctive Programming: Cutting Planes from Logical Conditions", in: Mangasarian O.L. et al., eds., *Nonlinear Programming*, Vol. 2, Academic Press, New York (1975) 279–312.

[3] Bixby R.E., Ceria S., McZeal C.M., Savelsbergh M.W.P, MIPLIB 3.0, www.caam.rice.edu/~bixby/miplib/miplib.html.

[4] Cohen P.R., *Empirical Methods for Artificial Intelligence*, MIT Press (1995).

[5] COIN-OR, www.coin-or.org.

[6] Cook W., Dash S., Fukasawa R., Goycoolea M., "Numerically Accurate Gomory Mixed-Integer Cut", preprint (2007).

[7] Dhiflaoui M., Funke S., Kwappik C., Mehlhorn K., Seel M., Sch¨omer E., Schulte R., Weber D., "Certifying and Repairing Solutions to Large LPs. How Good are LP-solvers?", Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003), 255–256, ACM, New York, (2003).

[8] Gomory R., "An Algorithm for the Mixed Integer Problem", Technical Report RM-2597, The RAND Corporation (1960).

[9] Hooker J.N., "Needed: An Empirical Science of Algorithms", *Operations Research* 42 (1994), 201–212.

[10] Hooker J.N., "Testing Heuristics: We Have It AllWrong", *Journal of Heuristics* 1 (1995), 33–42.

[11] Jeroslow R., "Cutting Plane Theory: Disjunctive Methods", Annals of Discrete Mathematics 1 (1972) 293–330.

[12] Koch T., "The Final Netlib Results", *Operations Research Letters* 32 (2004), 138–142.

[13] Marchand H., Martin A., Weismantel R., Wolsey L., "Cutting planes in integer and mixed integer programming", Workshop on Discrete Optimization, DO'99 (Piscataway, NJ), *Discrete Appl. Math.* 123 (2002), 397–446.

[14] F. Margot, "Testing Cut Generators for MILP", TepperWorking Paper 2007-E43 (2007).

[15] McGeoch C.C., "Toward an Experimental Method for Algorithm Simulation", INFORMS *Journal on Computing* 8 (1996), 1–15.

[16] McGeoch C.C., "Experimental Analysis of Algorithms", *Notices of the American Mathematical Association* 48 (2001), 304–311.

[17] McGeoch C.C., "Experimental Analysis of Optimization Algorithms", *Handbook of applied optimization*, Oxford University Press (2002), 1044–1052.

[18] McGeoch C.C., "Experimental Analysis of Algorithms", *Handbook of global optimization*, Vol. 2, Kluwer (2002), 489–513.

[19] Nemhauser G.L., Wolsey L.A., "A Recursive Procedure to Generate all Cuts for 0-1 Mixed Integer Programs", *Mathematical Programming* 46 (1990) 379–390.

[20] Neumaier A., Shscherbina O., "Safe Bounds in Linear and Mixed-Integer Linear Programming", *Mathematical Programming* 99 (2004), 283–296.

[21] R statistical software, www.r-project.org

# MPS Chair's Column — Optima 77

**Steve Wright**

3 June 2008

The summer conference season is already in full swing as I write. IPCO 2008 took place last week in Bertinoro, and from all reports was another outstanding success. Our thanks go to Andrea Lodi and Alessandro Panconesi and the other members of the organizing committee and to Giovanni Rinaldi and the program committee, and of course to the attendees too, for maintaining the high standards of this event. Another meeting organized in association with MPS – EngOpt 2008 – is taking place this week in Rio de Janeiro.

Many MPS members, including many from Europe, were in attendance at the SIAM Conference on Optimization in Boston last month. The conference had excellent plenary presentations and many other highlights, including technical sessions by students and collaborators of Society stalwart Mike Todd, who celebrated his 60th birthday during ICCOPT in 2007. A dinner honoring Mike along with Dave Shanno, also a distinguished long-time member of MPS, followed the conference.

I am delighted to announce that Jon Lee (IBM) has agreed to become Chair of the Executive Committee, the senior appointed position in MPS. In this role, Jon becomes an ex-officio member of all committees of the Society. We'll much appreciate his advice on the many important issues facing the society in the months ahead.

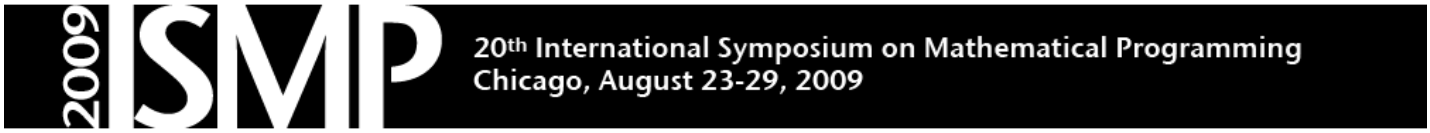
Following the great success of the first two ICCOPTs (Rensselaer, 2004 and McMaster, 2007), a steering committee headed by Tamas Terlaky solicited proposals for the third meeting in the series, planned for 2010. The committee evaluated several excellent alternatives, and eventually recommended a proposal to hold the meeting at the University of Chile in Santiago, July 2010. There is excellent support from Chilean institutions for the meeting and for the Winter School that precedes it. (Note that July is winter in Santiago – though surely an infinitely milder winter than the one we just experienced in Madison!) I am grateful to all those who submitted proposals and to Tamas and his committee for their hard work. We look forward to helping conference chair Alejandro Jofre and his colleagues in Chile in any way we can to make ICCOPT III as successful as earlier editions of this conference.

Organization of ISMP XX (Chicago, August 23-29) is proceeding apace, with the opening session scheduled for Symphony Center (home of the renowned Chicago Symphony Orchestra) and the banquet to be held at the Field Museum, Chicago's famous natural history museum. Meeting themes and plenary speakers will be announced soon. Please keep an eye on www.ismp2009.org for more information as it becomes available.

As always, ISMP is the occasion on which our Society awards its prizes for professional excellence. The process of assembling committees for these prizes in well advanced. The society web site www.mathprog.org contains information about the prizes and calls for the 2009 awards. Please give some thought to putting forward worthy candidates from among your colleagues.

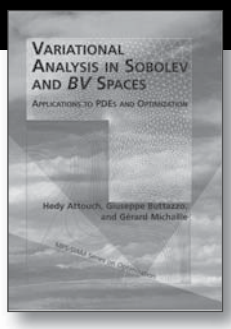
A niece of the late Ray Fulkerson – namesake of the Fulkerson prize – contacted me recently. She and other family members were previously not aware of the award, which dates to 1979, but were delighted to hear about it and are keen to participate in the 2009 award ceremony.

---




The 20th International Symposium on Mathematical Programming takes place August 23-29, 2009 in Chicago, Illinois and marks the 60th anniversary of the Zeroth Symposium organized by the Cowles Commission at the University of Chicago in June, 1949. The symposium will be held at the University of Chicago's Gleacher Center and the Marriott Downtown Chicago Magnificent Mile Hotel. Festivities include the opening session in Chicago's Orchestra Hall, home of the Chicago Symphony Orchestra, the conference banquet at the Field Museum, Chicago's landmark natural history museum, and a celebration of the 60th anniversary of the Zeroth Symposium. Plenary speakers and information on proposals for mini-symposia will be announced shortly on the conference Web site, www.ismp2009.org.

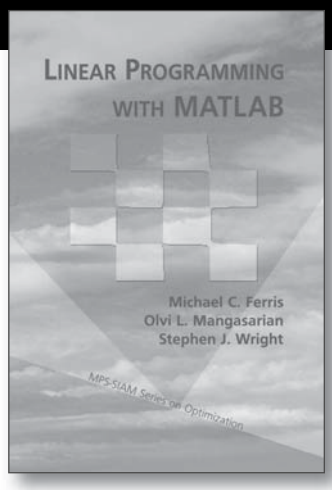
# *Call for Manuscripts*

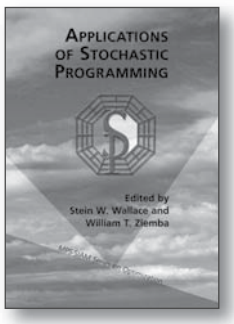



## MPS-SIAM Book Series on **OPTIMIZATION**

**Philippe Toint, Editor-in-Chief**
*University of Namur, Belgium*

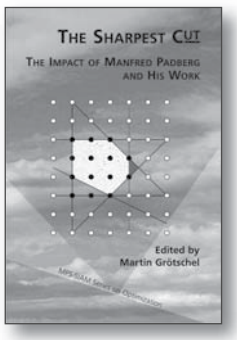
**The goal of the series is to publish a broad range of titles in the field of optimization and mathematical programming, characterized by the highest scientific quality.**



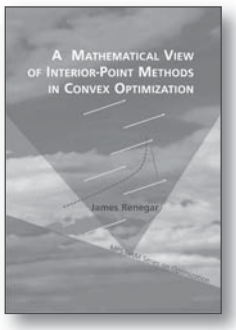

### BOOKS IN THE SERIES INCLUDE:

**Linear Programming with MATLAB**
Michael C. Ferris, Olvi L. Mangasarian, and Stephen J. Wright
2007 · xii + 266 pages · Softcover · ISBN 978-0-898716-43-6
List Price $45.00 · SIAM Member Price $31.50 · **Order Code MP07**




**Variational Analysis in Sobolev and *BV* Spaces: *Applications to PDEs and Optimization***
Hedy Attouch, Giuseppe Buttazzo, and Gérard Michaille
2005 · xii + 634 pages · Softcover · ISBN 978-0-898716-00-9
List Price $140.00 · MPS/SIAM Member Price $98.00 · **Order Code MP06**

**Applications of Stochastic Programming**
Edited by Stein W. Wallace and William T. Ziemba
2005 · xvi + 709 pages · Softcover · ISBN 978-0-898715-55-2
List Price $142.00 · MPS/SIAM Member Price $99.40 · **Order Code MP05**




**The Sharpest Cut: *The Impact of Manfred Padberg and His Work***
Edited by Martin Grötschel
2004 · xi + 380 pages · Hardcover · ISBN 978-0-898715-52-1
List Price $106.00 · MPS/SIAM Member Price $74.20 · **Order Code MP04**

**A Mathematical View of Interior-Point Methods in Convex Optimization**
James Renegar
2001 · viii + 117 pages · Softcover · ISBN 978-0-898715-02-6
List Price $47.00 · MPS/SIAM Member Price $32.90 · **Order Code MP03**

**YOU ARE INVITED TO CONTRIBUTE**

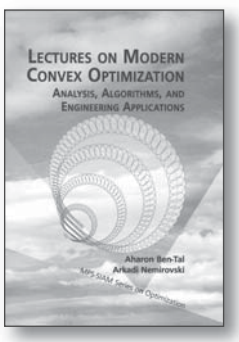
If you are interested in submitting a proposal or manuscript for publication in the series or would like additional information, please contact:

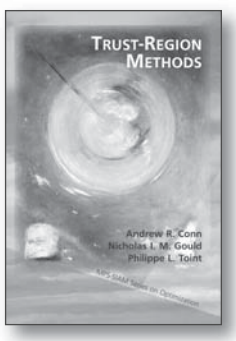
Philippe Toint
University of Namur
*philippe.toint@fundp.ac.be*

**OR**

Sara J. Murphy
Series Acquisitions Editor
SIAM
*murphy@siam.org*

SIAM publishes quality books with practical implementation at prices affordable to individuals.




**Lectures on Modern Convex Optimization: *Analysis, Algorithms, and Engineering Applications***
Aharon Ben-Tal and Arkadi Nemirovski
2001 · xvi + 488 pages · Softcover · ISBN 978-0-898714-91-3
List Price $121.50 · MPS/SIAM Member Price $85.05 · **Order Code MP02**

**Trust-Region Methods**
A. R. Conn, N. I. M. Gould, and Ph. L. Toint
2000 · xx + 959 pages · Hardcover · ISBN 978-0-898714-60-9
List Price $146.50 · MPS/SIAM Member Price $102.55 · **Order Code MP01**

siam®

**Complete information about SIAM and its book program can be found at *www.siam.org/books*. See summaries, tables of contents, and order online at *www.siam.org/catalog*.**

O P T I M A

MATHEMATICAL PROGRAMMING SOCIETY

UF | UNIVERSITY of FLORIDA

Center for Applied Optimization
401 Weil Hall
PO Box 116595
Gainesville, FL 32611-6595 USA

FIRST CLASS MAIL



PUBLISHED BY THE
MATHEMATICAL PROGRAMMING SOCIETY &
University of Florida