N°72

OPTIMA

Mathematical Programming Society Newsletter

JANUARY 2006

72

Local Branching: Basics and Extensions 2 Prizes 6 gallimaufry 9

Local Branching: Basics and Extensions

Matteo Fischetti*
Andrea Lodi°†

**DEI, University of Padova*
Via Gradenigo 6/A, 35100 Padova, Italy

°DEIS, University of Bologna
Viale Risorgimento 2, 40136 Bologna, Italy

†IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598

E-mail: matteo.fischetti@unipd.it
alodi@us.ibm.com

July 5, 2005

Abstract

The availability of effective exact or heuristic solution methods for general Mixed-Integer Programs (MIPs) is of paramount importance for practical applications. In the present paper we investigate the use of a generic MIP solver as a black-box "tactical" tool to explore effectively suitable solution subspaces defined and controlled at a "strategic" level by a simple external branching framework. The procedure is in the spirit of well-known local search metaheuristics, but the neighborhoods are obtained through the introduction in the MIP model of completely general linear inequalities called *local branching* cuts. The new solution strategy is exact in nature, though it is designed to improve the heuristic behavior of the MIP solver at hand. It alternates high-level strategic branchings to define the solution neighborhoods and low-level tactical branchings to explore them. The result is a completely general scheme aimed at favoring early updatings of the incumbent solution, hence producing high-quality solutions at early stages of the computation.

1 Introduction

Mixed-Integer linear Programming (MIP) plays a central role in modeling difficult-to-solve (NP-hard) combinatorial problems. However, the exact solution of the resulting models often cannot be carried out for the problem sizes of interest in real-world applications; hence, one is interested in effective heuristic methods. Although several heuristic and metaheuristic frameworks have been proposed in the literature for specific classes of problems, only a few papers deal with general-purpose MIP heuristics, including [1], [8], [9], [11], [12], [14] and [3] among others.

Exact MIP solvers are nowadays very sophisticated tools designed to hopefully deliver, within acceptable computing time, a provable optimal solution of the input MIP model, or at least a heuristic solution with a practically-acceptable error. In fact, what matters in many practical cases is the possibility of finding reasonable solutions as early as possible during the computation. In this respect, the "heuristic behavior" of the MIP solver plays a very important role: An aggressive solution strategy that improves the incumbent solution at very early stages of the computation is strongly preferred to a strategy designed for finding good solutions only at the late steps of the computation (that for difficult problems will unlikely be reached within the time limit). Many commercial MIP solvers allow the user to have a certain control on their heuristic behavior through a set of parameters affecting the visit of the branching tree, the frequency of application of the internal heuristics, the fact of emphasizing the solution integrality rather than its optimality, etc. Unfortunately, in some hard cases a general-purpose MIP solver may prove not adequate even after a clever tuning, and one tends to quit the MIP framework and to design ad-hoc heuristics for the specific problem at hand, thus losing the advantage of working in a generic framework.

Recently, Fischetti and Lodi [5] investigated the use of a general-purpose MIP solver as a black-box "tactical" tool to explore effective suitable solution subspaces defined and controlled at a "strategic" level by a simple external branching framework. The procedure is in the spirit of well-known local search metaheuristics, but the neighborhoods are obtained through the introduction in the MIP model of (invalid) linear inequalities called *local branching* cuts. This allows one to work within a perfectly general MIP framework, and to take advantage of the impressive research and implementation effort devoted to the design of MIP solvers.

A main point: Soft vs. hard variable fixing.

A commonly used, and often effective, heuristic scheme fitting into the framework described in the introduction is based on the so-called *(hard) variable fixing* or *diving* idea, which can be described as follows. We assume to have an exact or heuristic black-box solver for the problem at hand. The solver is first applied to the input data, but its parameters are set so as to quickly abort execution and return a (possibly infeasible) "target solution" $\overline{x}$. This solution is defined, e.g., as the solution of the root-node Linear Programming (LP) relaxation, possibly after a clever rounding of some of its fractional variables, or as any heuristic solution of the problem. Solution $\overline{x}$ is then analyzed, and some of its nonzero variables

are heuristically rounded up to the nearest integer (if non-integer) and then fixed to this value. The method is then iteratively re-applied on the restricted problem resulting from fixing: The black-box solver is called again, a new target solution is found, some of its variables are fixed, and so on. In this way the problem size reduces after each fixing, hence the black-box solver can concentrate on smaller and smaller "core problems" with increasing chances of solving them to proven optimality.

A critical issue in variable-fixing methods is of course related to the choice of the variables to be fixed at each step. As a matter of fact, for difficult problems high-quality solutions are only found after several rounds of fixing. On the other hand, wrong choices at early fixing levels are typically very difficult to detect, even when bounds on the optimal solution value are computed before each fixing: In the hard cases, the bound typically remains almost unchanged for several fixings, and increases suddenly after an apparently-innocent late fixing. Therefore, one has to embed in the scheme backtracking mechanisms to recover from bad fixings, a very difficult task.

The question is then how to fix a relevant number of variables without losing the possibility of finding good feasible solutions. To better illustrate this point, suppose one is given a heuristic 0-1 solution $\bar{x}$ of a pure 0-1 MIP model with n variables, and wants to concentrate on a core subproblem resulting from fixing to 1 at least 90% (say) of its nonzero variables. How should one choose the actual variables to be fixed? Put in these terms, the question lends itself to a simple answer: Just add to the MIP model a linear *soft fixing* constraint of the form

$$\sum_{j=1}^{n} \bar{x}_j x_j \geq \left\lceil 0.9 \sum_{j=1}^{n} \bar{x}_j \right\rceil \qquad (1)$$

and apply the black-box solver to the resulting MIP model. In this way one avoids a too-rigid fixing of the variables in favor of a more flexible condition defining a suitable neighborhood of the current target solution, to be explored by the black-box solver itself. In the example, the underlying hypothesis is that the 10% of slack left in the right-hand side of (1) drives the black-box solver as effectively as fixing a large number of variables, but with a much larger degree of freedom– hence better solutions can be found.

Good results using a hard fixing policy have been reported recently by Danna, Le Pape and Rothberg [3]. Their method called *Relaxation Induced Neighborhood Search* hard fixes in some nodes of a branch-and-cut tree each integer variable x_j that assumes an integer value in the current continuous relaxation, which is in turn coincident with its value in the incumbent solution. An eventually smaller additional MIP is then defined and heuristically explored in the attempt of improving the incumbent solution. The method appears in the arsenal of ILOG-Cplex since the recent version 9.0.

2 Local branching as a heuristic for MIPs

The soft fixing mechanism outlined in the previous section leads naturally to the general framework exploited in [5] to find good approximate solutions for hard MIPs. This framework, which is exact in nature, is designed to improve the heuristic behavior of the MIP solver at hand. It alternates high-level strategic branchings to define solution neighborhoods and low-level tactical branchings (performed within the MIP solver) to explore them. The result can then be viewed as a two-level branching strategy aimed at favoring early updatings of the incumbent solution, hence producing improved solutions at early stages of the computation. More precisely, we consider a generic MIP with 0-1 variables of the form:

$$(P) \qquad \min c^T x \qquad (2)$$

$$Ax \geq b \qquad (3)$$

$$x_j \in \{0,1\} \qquad \forall j \in \mathcal{B} \neq \emptyset \qquad (4)$$

$$x_j \geq 0, \text{integer} \qquad \forall j \in \mathcal{G} \qquad (5)$$

$$x_j \geq 0 \qquad \forall j \in \mathcal{C} \qquad (6)$$

Here, the variable index set $\mathcal{N}$: = {1, . . . , n} is partitioned into ($\mathcal{B}$, $\mathcal{G}$, $\mathcal{C}$), where $\mathcal{B} \neq \emptyset$ is the index set of the 0-1 variables, while the possibly empty sets $\mathcal{G}$ and $\mathcal{C}$ index the general integer and the continuous variables, respectively.

Given a feasible *reference solution* $\bar{x}$ of (P), let $\bar{S}$: = {$j \in \mathcal{B}$: = $\bar{x}_j$ = 1} denote the binary support of $\bar{x}$. For a given positive integer parameter k, we define the *k-OPT neighborhood* $\mathcal{N}(\bar{x}, k)$ of x as the set of the feasible solutions of (P) satisfying the additional *local branching constraint*:

$$\Delta(x, \bar{x}) := \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{S}} x_j \leq k \qquad (7)$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{x}$) either from 1 to 0 or from 0 to 1, respectively.

In the relevant case in which the cardinality of the binary support of any feasible solution of (P) is a constant, this constraint can more conveniently be written in its equivalent "asymmetric" form:

$$\sum_{j \in \bar{S}} (1 - x_j) \leq k' \; (= k / 2) \qquad (8)$$

The above definition is consistent, e.g., with the classical k'-OPT neighborhood for the Traveling Salesman Problem (TSP), where constraint (8) allows one to replace at most k' edges of the reference tour $\bar{x}$.

As its name suggests, the local branching constraint can be used as a branching criterion within an enumerative scheme for (P). Indeed, given the incumbent solution $\bar{x}$, the solution space associated with the current branching node can be partitioned by means of the disjunction:

$$\Delta(x, \bar{x}) \leq k \qquad \text{(left branch)}$$

or

$$\Delta(x, \bar{x}) \geq k + 1 \quad \text{(right branch)} \qquad (9)$$

As to the neighborhood-size parameter k, it should be chosen as the largest value producing a left-branch subproblem which is likely to be much easier to solve than the one associated with its father. The idea is that the neighborhood $\mathcal{N}(\bar{x}, k)$ corresponding to the left branch must be "sufficiently small" to be optimized within short computing time, but still "large enough" to likely contain better solutions than $\bar{x}$. According to our computational experience, the choice of k is seldom a problem by itself, in that values of k in range [10, 20] proved effective in most cases.

A first implementation of the local branching idea is illustrated in Figure 1, where the triangles marked by the letter "T" (for Tactical) correspond to the branching subtrees to be explored

through a standard "tactical" branching criterion such as, branching on fractional variables–i.e., they represent the application of the black-box exact MIP solver.

In the figure, we assume to have a starting incumbent solution $\bar{x}^1$ at the root node 1. The left-branch node 2 corresponds to the optimization within the k-OPT neighborhood $\mathcal{N}(\bar{x}^1, k)$, which is performed through a tactical branching scheme converging (hopefully in short computing time) to an optimal solution in the neighborhood, say $\bar{x}^2$. This solution then becomes the new incumbent solution. The scheme is then re-applied to the right-branch node 3, where the exploration of $\mathcal{N}(\bar{x}^2, k) \setminus \mathcal{N}(\bar{x}^1, k)$ at node 4 produces a new incumbent solution $\bar{x}^3$. Node 5 is then addressed, which corresponds to the initial problem (P) amended by the two additional constraints $\Delta(x, \bar{x}^1) \geq k + 1$ and $\Delta(x, \bar{x}^2) \geq k + 1$. In the example, the left-branch node 6 produces a subproblem that contains no improving solution. In this situation the addition of the branching constraint $\Delta(x, \bar{x}^3) \geq k + 1$ leads to the right-branch node 7, which is explored by tactical branching. Note that the fractional LP solution of node 1 is not necessarily cut off in *both* son nodes 2 and 3, as is always the case when applying standard branching on variables. The same holds for nodes 3 and 5. In fact, the local branching philosophy is quite different from the standard one: We do not want to force the value of a fractional variable, but we rather instruct the solution method to explore first some promising regions of the solution space. The advantage of the local-branching scheme is an early (and more frequent) update of the incumbent solution. In other words, we quickly find better and better solutions until we reach a point where local branching cannot be applied any more (node 7, in the example); hence, we have to resort to tactical branching to conclude the enumeration.

This behavior is illustrated in Figure 2, where we solved MIP instance `tr24-15` [15] by means of three codes: The commercial solver ILOG-Cplex 7.0 in the two versions emphasizing the solution *optimality* or *feasibility*, respectively, and the local branching scheme where ILOG-Cplex 7.0 (optimality version) is used to explore the "T-triangle" subtrees, and the local branching constraints are of type (7) with $k = 18$. Apart from the emphasizing setting, all the three codes were run with the same parameters. As to the initial reference solution $\bar{x}^1$ needed in the local branching framework, it was obtained as the first feasible solution found by ILOG-Cplex 7.0 (optimality version)–the corresponding computing time is included in the local-branching running time. The test was performed on Digital Alpha Ultimate Workstation 533 MHz.

According to the figure, the performance of the local branching scheme is quite satisfactory in that it is able to improve the initial solution several times in the early part of the computation. As a matter of fact, the local-branching incumbent solution is significantly better than that of the two other codes during almost the entire run. As to optimality convergence, the local branching method concludes its run after 1,878 CPU seconds, whereas ILOG-Cplex 7.0 in its optimization version converges to optimality within 3,827 CPU seconds (the feasibility version is unable to prove optimality within a time limit of 6,000 CPU seconds). Note, however, that the enhanced convergence behavior of the local branching scheme in proving optimality cannot be guaranteed in all cases. Indeed, the framework described in Figure 1 has been specialized to obtain good approximate solutions for hard MIPs by using effective ideas from the metaheuristic area such as a time limit on the exploration of each node and sophisticated diversification strategies. For space limit, we do not include the final description of the framework (see [5] for details), but an example of the execution on the hard MIP instance `B1C1S1` [15] is reported in Figure 3. Complete results are given in [5].

3 Local branching extensions

The main idea discussed in Section 1 opens many interesting fields of application in which the basic local branching idea can extend its range of use.

Tighter integration within the MIP solver. There are two ways of exploiting local branching constraints. The first uses the local branching constraints as a "strategic" branching rule within an exact solution method, to be alternated with a more standard "tactical" branching rule. This approach, described in Section 2,

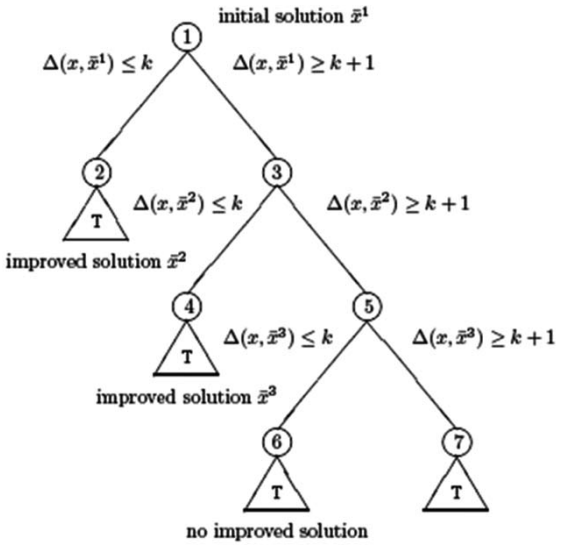


Figure 1: The basic local branching scheme.

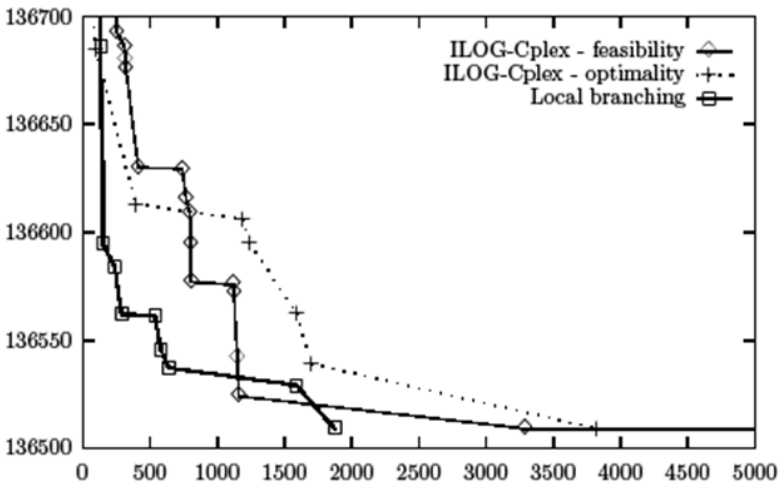


Figure 2: Solving MIP instance `tr24-15` (solution value vs. CPU seconds).

uses the MIP solver as a black-box for performing the tactical branchings. This is remarkably simple to implement, but has the disadvantage of wasting part of the computational effort devoted, to the exploration of the nodes where no improved solution could be found within the node time limit. Therefore, a more integrated (and flexible) framework where the two branching rules work in tight cooperation is expected to produce an enhanced performance.

Local search by branch-and-cut.

A second way of using the local branching constraints is within a genuine heuristic framework akin to *Tabu Search* (TS) or *Variable Neighborhood Search* (VNS). As a matter of fact, all the main ingredients of these metaheuristics (defining the current solution neighborhood, dealing with tabu solutions or moves, imposing a proper diversification, etc.) can easily be modeled in terms of linear cuts to be dynamically inserted and removed from the model. This naturally leads to a completely general (and hopefully powerful) TS or VNS framework for MIPs possibly designed to take into account the structure of specific combinatorial problems. Very promising results in this direction have been obtained by Fischetti, Polo and Scantamburlo [7].

Working with infeasible reference solutions.

As stated, the local branching framework requires a starting (feasible) reference solution $\bar{x}$[1], which we assume is provided by the MIP black-box solver. However, for difficult MIPs (such as, hard set partitioning models) even the definition of this solution may require an excessive computing time. In this case, one should consider the possibility of working with *infeasible* reference solutions. It is then advisable to adopt the widely-used mechanism in metaheuristic frameworks consisting in modifying the MIP model by adding slack variables to (some of) the constraints, while penalizing them in the objective function. A sophisticated version of this simple idea has been considered by Fischetti and Lodi [6] in conjunction with a recently proposed algorithm to find an initial solution for MIPs called *Feasibility Pump* [4].

Dealing with general-integer variables.

Local branching is based on the assumption that the MIP model contains a relevant set of 0-1 variables to be used in the definition of the "distance function" $\Delta(x, \bar{x})$. According to our computational experience, this definition is effective even in case of MIPs involving general integer variables, in that the 0-1 variables (which are often associated with big-M terms) are likely to be largely responsible for the difficulty of the model. However, it may be the case that the MIP model at hand does not involve any 0-1 variable, or that the 0-1 variables do not play a relevant role in the model–hence the introduction of the local branching constraint does not help the MIP solver. In this situation, one is interested in modified local branching constraints including the general-integer variables in the definition of the distance function $\Delta(x, \bar{x})$. To this end, suppose MIP model (P) involves the bounds $l_j \le x_j \le u_j$ for the integer variables x_j ($j \in \mathcal{I} := \mathcal{B} \cup \mathcal{G}$). Then a suitable local branching constraint can be defined as follows:

$$\Delta_1(x,\bar{x}) := \sum_{j\in I:\bar{x}_j=l_j} \mu_j(u_j - x_j) +$$

$$\sum_{j\in I:\bar{x}_j=u_j} \mu_j(u_j - x_j) +$$

$$\sum_{j\in I:\bar{x}_j<\bar{x}_j<u_j} \mu_j(x_j^+ - x_j^-) \le k$$

where weights μ_j are defined, e.g., as $\mu_j = 1/(u_j - l_j)$ for all $j \in \mathcal{I}$, while the variation terms x_j^+ and x_j^- require the introduction into the MIP model of additional constraints of the form:

$$x_j = \bar{x}_j + x_j^+ - x_j^-,\ \ x_j^+ \ge 0,\ \ x_j^- \ge 0,$$

$$\forall j \in \mathcal{I} : l_j < \bar{x}_j < u_j .$$

Use of local branching constraints within special-purpose codes.

As outlined in the introduction, there is no need of using local branching constraints within a general-purpose MIP solvers. In fact, these constraints can be effectively integrated within special-purpose (black-box) codes, both exact and heuristic, designed for specific problems so as to enhance their heuristic capability. Obviously the only requirement is that the code is able to take into account linear inequalities. In this context, using local branching constraints within a special-purpose branch-and-cut code seems to be very suitable, and interesting results have been obtained by Hernández-Pérez and Salazar [10].

4 On-line Resources

A beta version of the `LocBra` code is available for research purposes together with a collection of hard MIP instances, and some papers and slides illustrating the method at the web page: www.or.deis.unibo.it/research_pages/ORinstances/MIPs.html Finally, a Local Branching heuristic is now part of the arsenal of ILOG-Cplex since version 9.1.

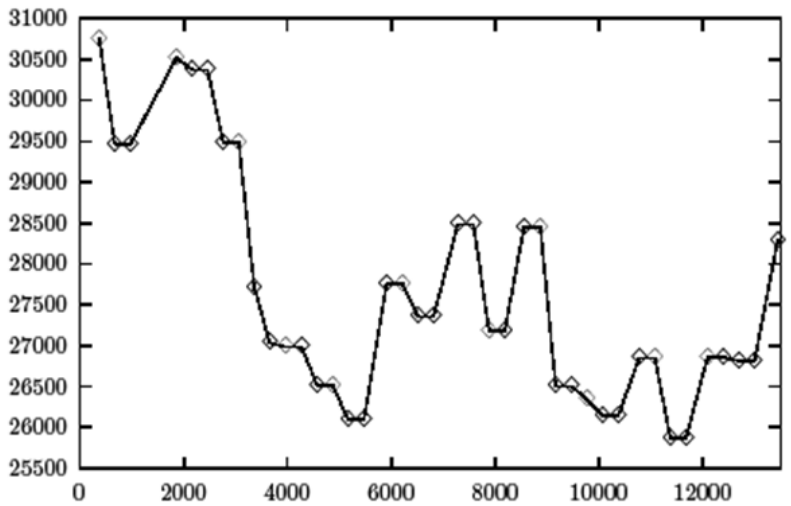


Figure 3: `LocBra` as a heuristic for instance `B1C1S1` (solution value vs. CPU sec.).

References

[1] E. Balas, S. Ceria, M. Dawande, F. Margot and G. Pataki. OCTANE: A New Heuristic For Pure 0-1 Programs. *Operations Research* 49(2), 207-225, 2001.

[2] Cplex. ILOG Cplex 7.0 *User's Manual and Reference Manual*. *ILOG*, S.A., 2001 (http://www.ilog.com)

[3] E. Danna, E. Rothberg, C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming* 102, 71-90, 2005.

[4] M. Fischetti, F. Glover, A. Lodi. The Feasibility Pump. *Mathematical Programming* DOI 10.1007/s10107-004-0570-3, 2005.

[5] M. Fischetti, A. Lodi. Local Branching. *Mathematical Programming* 98, 23-47, 2003.

[6] M. Fischetti, A. Lodi, "Repairing MIP Infeasibility through Local Branching", *Technical Report* RC23532, IBM T.J. Watson Research Center, 2005.

[7] M. Fischetti, C. Polo, M. Scantamburlo. A Local Branching Heuristic for Mixed-Integer Programs with 2-Level Variables. *Networks* 44, 2, 61-72, 2004.

[8] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part I. *Journal of Heuristics* 2, 343-358, 1997.

[9] F. Glover and M. Laguna. General Purpose Heuristics For Integer Programming: Part II. *Journal of Heuristics* 3, 161-179, 1997.

[10] H. Hernández-Pérez and J.J. Salazar-González. Heuristics for the one commodity Pickup-and-Delivery Traveling Salesman Problem. *Transportation Science*, 2003 (to appear).

[11] A. Løkketangen. Heuristics for 0-1Mixed-Integer Programming. In P.M. Pardalos and M.G.C. Resende (ed.s) *Handbook of Applied Optimization*, Oxford University Press, 474-477, 2002.

[12] A. Løkketangen and F. Glover. Solving Zero/One Mixed Integer Programming Problems Using Tabu Search. *European Journal of Operational Research* 106, 624-658, 1998.

[13] N. Mladenovíc and P. Hansen. Variable Neighborhood Search. *Computers and Operations Research* 24, 1097-1100, 1997.

[14] M. Nediak and J. Eckstein. *Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming*. Research Report RRR 53-2001, RUTCOR, Rutgers University, October 2001.

[15] M. Van Vyve and Y. Pochet. A General Heuristic for Production Planning Problems. *CORE Discussion Paper* 56, 2001.

DANTZIG PRIZE

Call for nominations 2006

Nominations are solicited for the George B. Dantzig Prize, administered jointly by the Mathematical Programming Society (MPS) and the Society for Industrial and Applied Mathematics (SIAM). This prize is awarded to one or more individuals for original research which by its originality, breadth and depth is having a major impact on the field of mathematical programming. The contribution(s) for which the award is made must be publicly available and may belong to any aspect of mathematical programming in its broadest sense. Preference will be given to candidates who have not reached their 50th birthday in the year of the award.

The prize will be presented at the 2006 SIAM Annual Meeting to be held July 10-14, 2006, in Boston, Mass. Past prize recipients are listed on the MPS Web site. The members of the prize committee are Robert Bixby (Chair), Arkadi Nemirovski, Jong-Shi Pang, and Alexander Schrijver.

Nominations should consist of a letter describing the nominee's qualifications for the prize and a current curriculum vitae of the nominee, including a list of publications. They should be sent to the chair of the committee and received by Nov. 15 2005. Submission of nomination materials in electronic form is strongly encouraged.

Robert E. Bixby
ILOG, Inc. and Rice University
8 Briarwood Ct.
Houston, Texas, 77019
USA

E-mail: bixby@ilog.com or bixby@rice.edu

SÉMINAIRE DE MATHÉMATIQUES

SUPÉRIEURES 2006/NATO ADVANCED STUDY INSTITUTE

45th session

Combinatorial Optimization:

Methods and Applications

JUNE 19-30 2006

Université de Montréal. Requests for participation or financial assistance must be received before Feb. 28 2006.

www.dms.umontreal.ca/sms/

The Lagrange Prize in Continuous Optimization

Call for Nominations

Nominations are invited for the Lagrange Prize in Continuous Optimization, awarded jointly by the Mathematical Programming Society (MPS) and the Society for Industrial and Applied Mathematics (SIAM). The prize will be presented at the SIAM Annual Meeting in July 2006.

To be eligible, works should form the final publication of the main result(s) and should be published either (a) as an article in a recognized journal or in a comparable, well-referenced volume intended to publish final publications only; or (b) as a monograph consisting chiefly of original results rather than previously published material. Extended abstracts and prepublications, and articles published in journals, journal sections, or proceedings that are intended to publish non-final papers, are not eligible. The work must have been published during the six calendar years preceding the year of the award meeting.

Judging of works will be based primarily on their mathematical quality, significance, and originality. Clarity and excellence of the exposition and the value of the work in practical applications may be considered as secondary attributes.

The Prize Committee for 2006 consists of John Dennis, Nick Gould, Adrian Lewis, and Mike Todd. Full details and prize rules are given at www.mathprog.org/prz/lagrange.htm To nominate a publication for the prize, please send a copy of the paper and a letter of nomination by January 23, 2006 to :

Michael J. Todd
School of Operations Research
Rhodes Hall
Cornell University
Ithaca, NY 14853-3801

E-mail: miketodd@cs.cornell.edu

Electronic submissions are preferred.

A. W. TUCKER PRIZE

Call for nominations 2006

The next A. W. Tucker Prize will be awarded at the XIXth International Symposium on Mathematical Programming in Rio de Janeiro, 30 July-4 August 2006, for an outstanding paper authored by a student.

The paper can deal with any area of mathematical programming. All students, graduate or undergraduate, are eligible. Nominations of undergraduate students are welcome. The Awards Committee will screen the nominations and select at most three finalists. The finalists will be invited, but not required, to give oral presentations at a special session of the symposium. The awards committee will select the winner before the symposium and present the award prior to the conclusion of the symposium.

The paper may be original research, a particularly notable exposition or survey, a report on innovative computer routines and computing experiments, or a presentation of a new and ingenious application. The paper must be solely authored and completed since 2003. A Ph.D. thesis qualifies. The paper and the work on which it is based should have been undertaken and completed in conjunction with a degree program.

Nominations must be made in writing to the chairman of the awards committee by a faculty member at the institution where the nominee was a student when the paper was completed. Moreover, nominators should send one copy each of: 1) the student's paper and a separate summary of the paper's contributions, written by the nominee, no more than two pages in length and 2) a brief biographical sketch of the nominee to each of the four members of the Tucker Prize committee:

Prof. S. Thomas McCormick (Chair)
Sauder School of Business
University of British Columbia
Vancouver, B.C., Canada V6T 1Z2
Tom.McCormick@sauder.ubc.ca

Prof. Jong-Shi Pang
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, New York 12180-3590, USA
pangj@rpi.edu

Prof. Monique Laurent
CWI, Centrum voor Wiskunde
en Informatica
Kruislaan 413
NL-1098 SJ Amsterdam, The Netherlands
M.Laurent@cwi.nl

Prof. Rüdiger Schultz
Department of Mathematics
University of Duisburg-Essen Campus
Lotharstr. 65
D-47057 Duisburg, Germany
schultz@math.uni-duisburg.de

The awards committee may request additional information.

The deadline for nominations is February 1, 2006.

Nominations and the accompanying documentation must be written in a language acceptable to the awards committee. The winner will receive an award of $750 (U.S.) and a certificate. The other finalists will also receive certificates. The Society will also pay partial travel expenses for each finalist to attend the symposium. These reimbursements will be limited in accordance with the amount of endowment income available. A limit in the range from $500 to $750 (U.S.) is likely. The institutions from which the nominations originate will be encouraged to assist any nominee selected as a finalist with additional travel expense reimbursement. Previous winners and further information about the Tucker Prize can be found at www.mathprog.org/prz/tucker.htm#winners.

D.R. FULKERSON PRIZE

Call for nominations

The Fulkerson Prize Committee invites nominations for the Delbert Ray Fulkerson Prize, sponsored jointly by the Mathematical Programming Society (MPS) and the American Mathematical Society. Up to three awards of US $1500 each are presented at each (triennial) International Symposium of the MPS. The Fulkerson Prize is for outstanding papers in the area of discrete mathematics. The prize will be awarded at the 19th International Symposium on Mathematical Programming to be held in Rio de Janeiro, Brazil from July 30 to Aug. 4, 2006.

Eligible papers should represent the final publication of the main result(s) and should have been published in a recognized journal, or in a comparable, well-refereed volume intended to publish final publications only, during the six calendar years preceding the year of the Symposium (thus, from Jan. 2000 through Dec. 2005). The prizes will be given for single papers, not series of papers or books, and in the event of joint authorship the prize will be divided. The term "discrete mathematics" is interpreted broadly and is intended to include graph theory, networks, mathematical programming, applied combinatorics, applications of discrete mathematics to computer science, and related subjects. While research work in these areas is usually not far removed from practical applications, the judging of papers will be based only on their mathematical quality and significance.

Further information about the Fulkerson Prize, including a list of previous winners, can be found at www.mathprog.org/prz/fulkerson.htm and at www.ams.org/prizes/fulkerson-prize.html.

The Fulkerson Prize committee consists of Noga Alon (Tel-Aviv University), Bill Cunningham (U. Waterloo) and Michel Goemans (MIT), chair.

Please send your nominations (including reference to the nominated article and an evaluation of the work) by Jan. 31, 2006 to the committee chair. Electronic submissions to goemans@math.mit.edu are preferred.

Michel Goemans
MIT, Room 2-351
Department of Mathematics
77 Massachusetts Avenue
Cambridge, MA 02139
USA

E-mail: goemans@math.mit.edu

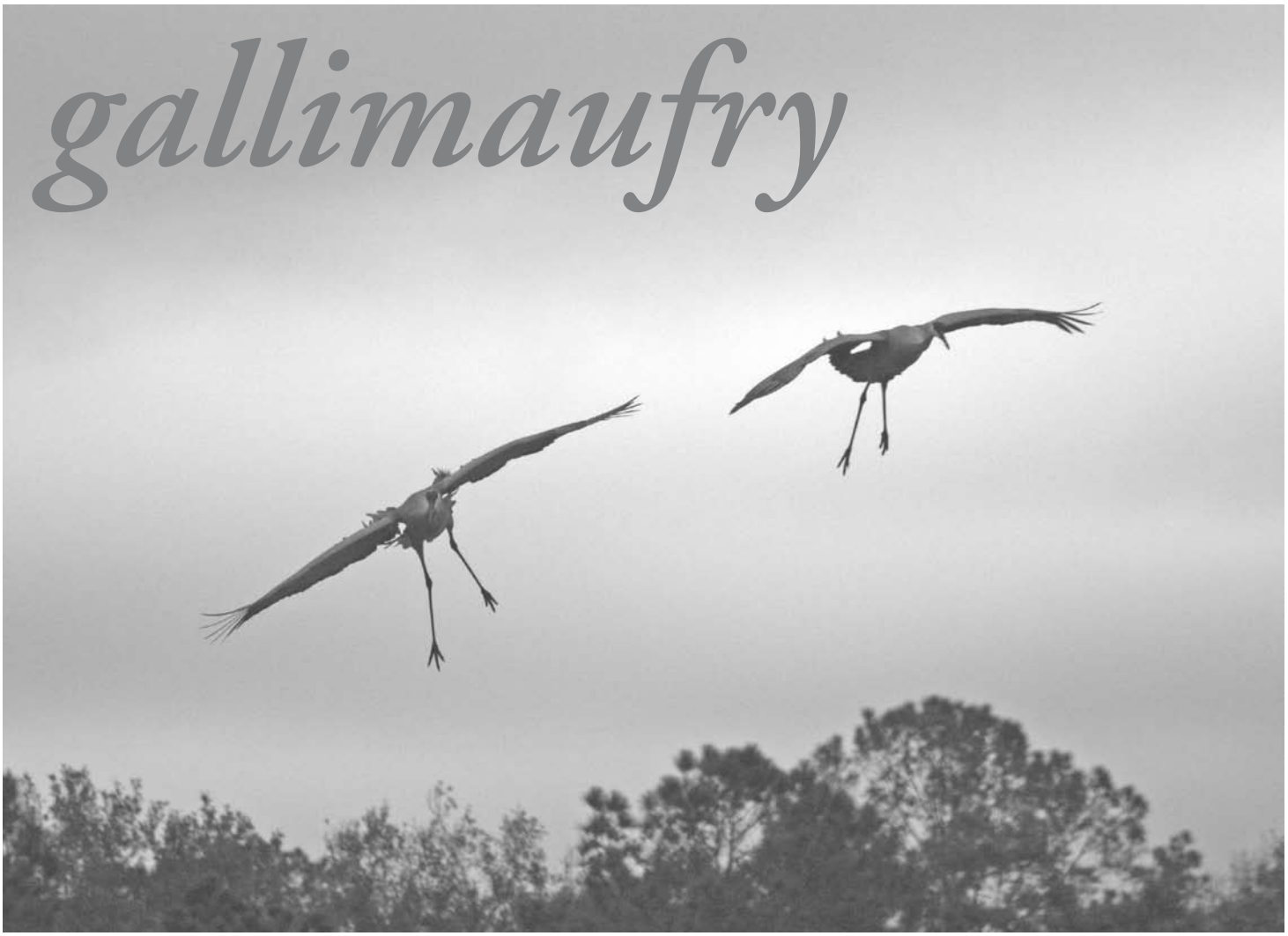
Beale-Orchard-Hays Prize

Call for nominations 2006

The Mathematical Programming Society invites nominations for the Beale-Orchard-Hayes Prize for Excellence in Computational Mathematical Programming. For details of rules and eligibility, please see www.mathprog.org/prz/boh.htm

Nominations can be submitted either electronically or in writing, but not a combination of the two. The nomination must include a cover letter with the title, authors, and publication details of the nominated paper or book. If submitted electronically, the final published version of the nominated publication should be attached to the message. If in writing, please send four copies of the paper or book. Supporting justification and any supplementary materials are welcome but not mandatory. The screening committee reserves the right to request further supporting materials from the nominees. The deadline for nominations is March 17, 2006. Nominations should be mailed to:

Stephen Wright
Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706 USA
E-mail: swright@cs.wisc.edu



Application for Membership

I wish to enroll as a member of the Society.

My subscription is for my personal use and not for the benefit of any library or institution.

☐ I will pay my membership dues on receipt of your invoice.

☐ I wish to pay by credit card (Master/Euro or Visa).

CREDIT CARD NO. _____ EXPIRATION DATE _____

FAMILY NAME _____

MAILING ADDRESS _____

_____

TELEPHONE NO. _____ TELEFAX NO. _____

E-MAIL _____

SIGNATURE ▶ _____

Mail to:

Mathematical Programming Society
3600 University City Sciences Center
Philadelphia, PA 19104-2688 USA

Cheques or money orders should be made payable to The Mathematical Programming Society, Inc. Dues for 2006, including subscription to the journal ***Mathematical Programming***, are US $80. Retired are $50. Student applications: Dues are $25. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status

Institution

O P T I M A
MATHEMATICAL PROGRAMMING SOCIETY

UNIVERSITY OF
FLORIDA

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

FIRST CLASS MAIL

PUBLISHED BY THE
MATHEMATICAL PROGRAMMING SOCIETY &
GATOREngineering, PUBLICATION SERVICES
University of Florida