ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Experiments with MINLP Branching Techniques

Sven Leyffer

Mathematics and Computer Science Division

Preprint ANL/MCS-P1734-0310

March 23, 2010

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Contents

Experiments with MINLP Branching Techniques

Sven Leyffer

Abstract. Mixed-integer nonlinear optimization problems arise in scientific and operational applications ranging from the reordering of nuclear fuel rods to the design of wireless networks. We present some novel mixed-integer nonlinear optimization applications and review existing solution techniques. We also describe some experiments with nonlinear branch-and-bound branching techniques that lead us to promote a tighter integration of nonlinear solvers into a general branch-and-cut framework.

1. Introduction and Background

Many scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the optimality of the final design. Mixed-integer nonlinear programming (MINLP) optimization problems combine the difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. MINLP is one of the most flexible modeling paradigms available; and an expanding body of researchers and practitioners, including computer scientists, engineers, economists, statisticians, and operations managers, are interested in solving large-scale MINLPs. Such problems can be expressed conveniently as

$$\underset{x,y}{\text{minimize}} \ f(x,y) \quad \text{subject to } c(x,y) \leq 0, \ x \in X, \ y \in Y \text{ integer}, \tag{1.1}$$

where x, y are the continuous and integer variables, respectively, and X, Y are polyhedral sets. The functions f, c are assumed to be twice continuously differentiable and possibly convex. Surveys of MINLP can be found in [22, 24, 23].

Given the generality and flexibility of the model, MINLPs have been proposed for many diverse and important applications. A small subset of these applications includes portfolio optimization [5, 29], design of water distribution networks [10, 30], block layout design in the manufacturing and service sectors [11], network design with queuing delay constraints [9], operational reloading of nuclear reactors [35], integrated design and control of chemical processes [21], blackout prevention for electrical power systems [6, 15], and minimizing of the environmental impact of utility plants [16].

New MINLP Applications in Computer Science. Mixed integer nonlinear programs are fast becoming prevalent on the research frontiers of computer science. For example, there are many emerging applications of MINLP in communications research. Problems in wireless bandwidth allocation [4, 36, 13], selective filtering [37, 38], network design topology [3, 12], and optical network performance optimization [17] can all be cast as MINLPs.

We have begun building a library, called DIWAL, of MINLP test problems from computer science applications; see http://wiki.mcs.anl.gov/NEOS/index.php/DIWAL. Current applications include the following:

- Nonlinear optimization of IEEE 802.11 mesh networks [13]: model formulated to plan and optimize IEEE 802.11 broadband access networks.
- Distributed optimization for data-optical networking [17]: model to jointly optimize optical networking provisioning and Internet protocol traffic engineering.

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694 and by the National Science Foundation, Division Computing and Communication Foundations, through grant 0830035.
Keywords: Nonlinear Programming, Mixed-Integer Nonlinear Programming.

- Energy provisioning and relay node placement for wireless sensor networks [28]: model formulated to determine the optimal placement of provisioned energy among local aggregation and forwarding nodes and relay nodes such that the two-tiered network lifetime is maximized.
- Capacity fairness for wireless mesh networks [25]: model to assign channels to user nodes and determine power of transmission for mesh routers in a wireless network.

In some cases, these applications require detailed reformulations to avoid or mitigate nonconvexities. Next, we review a particular solution method for MINLP, namely, branch-and-bound, and then present ideas on how to improve this approach through a tighter integration of the MIP and NLP solves.

2. Nonlinear Branch-and-Bound

Nonlinear branch-and-bound dates back to the 1960s [31, 14]. It is best explained as a tree search. Initially, all integer restrictions are relaxed and the resulting nonlinear programming (NLP) relaxation is solved. Let the solution be $(\hat{x}, \hat{y})$. If all integer variables, $\hat{y}$, are integral, then we have solved the MINLP. Otherwise, we can choose some nonintegral integer to branch on. Branching on, say y_i, is achieved by creating two new NLP problems with added bounds $y_i \leq [\hat{y}_i]$ and $y_i \geq [\hat{y}_i] + 1$, respectively (where $[a]$ is the largest integer not greater than a). Next, one of these two NLPs is selected and solved, and the process is repeated. We can declare that a node has been fathomed if one of the following conditions is satisfied:

(1) An infeasible NLP is detected, implying that the whole subtree is infeasible.
(2) An integer feasible node is detected, which provides an upper bound on the optimum of the MINLP.
(3) A lower bound on the NLP solution is greater than or equal to the current upper bound, which implies that we cannot find a better solution in this subtree.

After a node has been fathomed, the algorithm backtracks to another open node until all nodes are fathomed. Heuristics for selecting a branching variable and nodes are discussed in [26, 39].

Typically, every NLP is solved from a previously saved primal-dual solution. In mixed-integer linear programming (MILP) it is sufficient to save a basis because a basis uniquely determines a primal-dual iterate for a linear program (LP). This situation does not generalize to MINLPs, however. Given a basis (or active set) is not sufficient to determine a starting point because the Jacobian also depends on the value of the variables, (x, y). In this paper we focus on a closer integration of the NLP solver and branch-and-bound, concentrating on one particular branching rule that has proved to be successful in MILP, namely, strong branching [2].

2.1. Preliminary Experience with Nonlinear Branch-and-Bound

We present some preliminary numerical results that motivate our interest in nonlinear branch-and-bound. We start by noting that MINLPBB [18] is typically outperformed by more modern approaches such as LP/NLP-based branch-and-bound [34, 8, 32, 1]. Figure 1 shows a performance profile of several MINLP solvers on a set of medium-sized problems. A performance profile can be interpreted as the probability distribution that a solver is at worse 2^x times worst than the best solver. Solvers whose lines are toward the left top are best.

We note that MINLPBB is a fairly simplistic nonlinear branch-and-bound solver. It implements a depth-first tree search with maximum fractional branching, which has been shown to be notoriously poor. Strong branching is usually superior to maximum fractional branching for solving MILPs [2]. We can readily generalize strong branching to MINLP. Given a solution of parent node NLP, P, with optimum value f^p, we perform the following steps:

(1) Find all *nonintegral* integer variables $y_i, i \in C$.
(2) For every candidate $y_i \in C$ solve *two child NLPs*:
    - A down NLP: $P \cup \{y_i = \lfloor y_i \rfloor\}$ with optimal value f_i^-.
    - An up NLP: $P \cup \{y_i = \lfloor y_i \rfloor + 1\}$ with optimal value f_i^+.

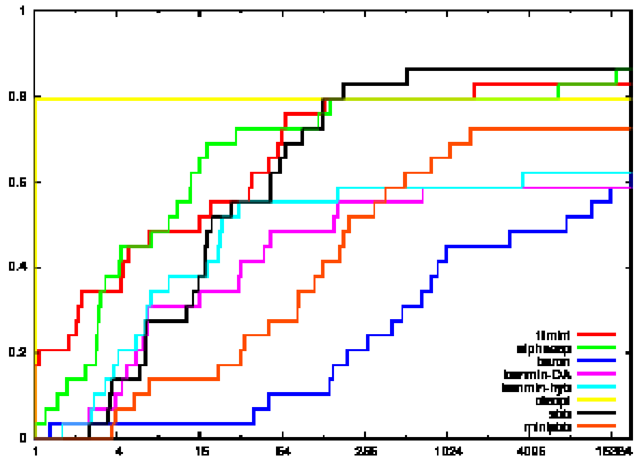

FIGURE 1. Performance profile of CPU time of several MINLP solvers on a set of medium-sized problems.

(3) For every candidate $y_i \in C$ compute its score:

$$\text{score}_i := (1-\mu)\min(f_i^- - f^p,\ f_i^+ - f^p)\ +\ \mu\max(f_i^- - f^p,\ f_i^+ - f^p),$$

where $\mu = 1/6$.

(4) Branch on the variable y_i that maximizes score_i.

The goal of this procedure is to maximize the change in the objective and select branching variables that change the problem the most [2].

Figure 2 shows the effect of strong branching for nonlinear branch-and-bound. The number of nodes in the tree is reduced significantly compared to maximum-fractional branching. However, the additional CPU time needed to solve these NLPs, even using SQP warm-starts, is still prohibitive, and strong branching is outperformed even by maximum fractional branching. The plots also show pseudo-cost branching, which outperforms both other options.

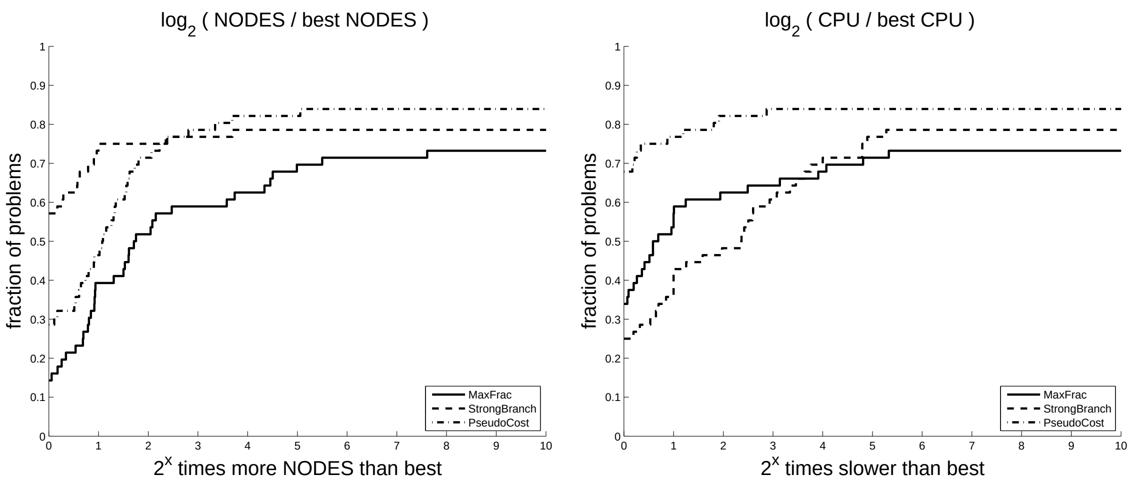

FIGURE 2. Performance profile comparing strong branching and maximum-fractional branching. The left plot shows the number of nodes; the right shows the CPU time.

Motivated by these observations, we next consider a closer integration of the NLP solver with nonlinear branch-and-bound to reduce the CPU time required for strong branching.

2.2. **Challenges in Integrating NLP and MIP**

In NLP, we cannot generate a vertex or primal-dual solution simply from a knowledge of the basis, or active set. The reason is that even given an optimal active set, we still need to solve a nonlinear problem (using, e.g., Newton's method) to obtain its solution, whereas in LP we simply update basis factors and perform a forward and a backward solve with the basis.

In principle, NLP solvers also compute factors that could be reused. Unfortunately, these factors are always outdated after a solve. To see why, consider a simple Newton iteration. At iteration k, we factor the Jacobian matrix, and compute a step, $z_{k+1} = z_k + d$. If z_{k+1} satisfies our stopping criterion, then we exit the solver without forming new factors. This situation is exacerbated in NLP, where not only do we have outdated factors, but the convergence test requires us to update the gradients (i.e Jacobian), so that factors and the stored matrices are out of sync after an NLP solve.

3. **Integrating NLP and MIP**

Our NLP solver is a sequential quadratic programming (SQP) method; see [27, 33, 7]. SQP methods successively minimize a quadratic model, $m_k(x)$, subject to a linearization of the constraints about $z_k = (x_k, y_k)$. We define the displacement $d := z - z_k$ and obtain the QP

$$\underset{d}{\text{minimize}}\ m_k(d) := g_k^T d + \frac{1}{2} d^T H_k d \quad \text{subject to } c_k + A_k^T d \le 0, \tag{3.1}$$

where $g_k = \nabla f(x_k, y_k)$ is the objective gradient, $c_k = c(x_k, y_k)$ are the values of the constraints, $A_k = \nabla c(x_k, y_k)$ is the Jacobian matrix, $H_k \simeq \nabla^2 L(z_k, \lambda_k)$ approximates the Hessian of the Lagrangian, and λ_k is the multiplier estimate at iteration k. The new iterate is $z_{k+1} = z_k + d$, together with the multipliers λ_{k+1} of the linearized constraints of (3.1).

We use the SQP solver FilterSQP [19] which implements a trust-region SQP method. Convergence is enforced with a filter [20], whose components are the ℓ_1-norm of the constraint violation, and the objective function.

We can improve strong branching in two ways. The first is to replace the costly NLP solve for every problem on the list of candidates branching variables, C, by a single QP solve. The second approach is to reuse as much of the final QP as possible solve from the previous iteration.

3.1. **Approximate Strong Branching**

The simplest way to improve strong branching is by replacing a complete NLP solve by a single iteration of SQP. Recall that we have already solved the parent problem, so we have a reasonable approximation of the solution that we obtained if we branched on one variable. This approach is readily implemented. Because the Hessian, H_k, and the Jacobian, A_k are outdated, however, we cannot readily reuse their factors (which are available after a solve with FilterSQP) and, instead, can only perform a warm-start in which we send the final optimal active set to the QP solver. We refer to this kind of branching as approximate strong branching. Special care has to be taken because every solve is only an approximate NLP, so the usual fathoming rules during strong branching have to be adapted.

Our preliminary numerical results in Figure 3 show that approximate strong branching (black line) improves on strong branching and is almost competitive with the simpler pseudo-cost branching, in terms of both number of nodes and CPU time.

We can improve our branching decisions further by adapting reliability branching to NLP. Reliability branching computes pseudo-cost estimates by strong branching until the resulting pseudo-cost estimate is deemed sufficiently reliable (measured by the number of times pseudo-costs have been updated for each integer variable). We use a threshold of 2 in our experiments, and we apply only approximate strong branching, rather than complete NLP solves. The results are displayed in Figure 4.

From Figure 4 we see that reliability branching is the method of choice for MINLP. Most important, reliability branching also outperforms BONMIN-Hybrid [8] in terms of CPU time. The comparison in terms of problems is less relevant because BONMIN-Hybrid counts only NLP solves

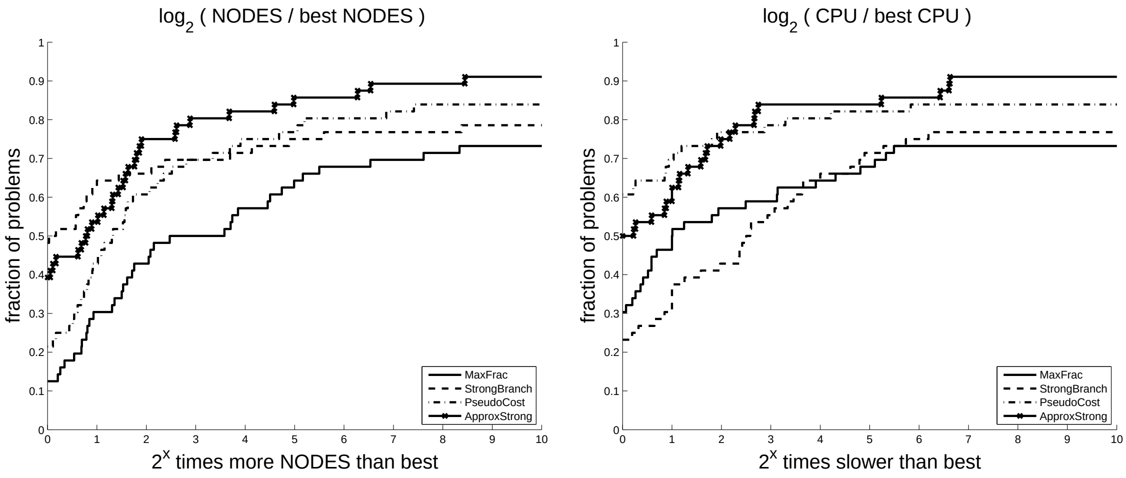

Figure 3. Performance profile comparing approximate strong branching with strong branching, maximum-fractional branching, and pseudo-cost branching. The left plot shows the number of nodes; the right shows the CPU time.

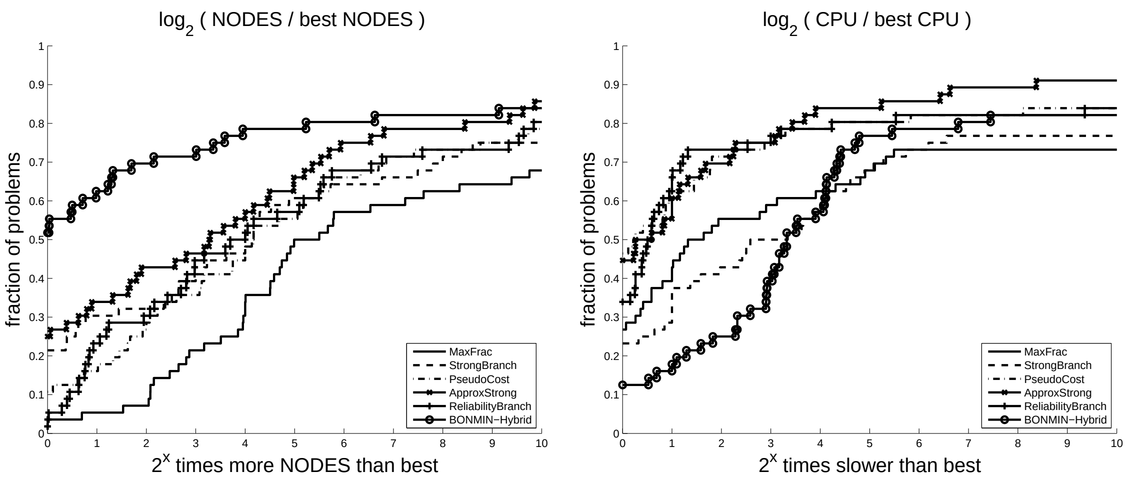

Figure 4. Performance profile comparing approximate strong branching with strong branching, maximum-fractional branching, pseudo-cost branching, reliability branching, and BONMIN-Hybrid. The left plot shows the number of nodes; the right shows the CPU time.

in this category, ignoring LP nodes that are solved. We are currently investigating the optimal choice of the reliability parameters for MINLP.

Next, we present an approach that allows us to reuse the factors of the final QP solve, in an attempt to gain further performance advances.

3.2. **Hot-Starting QP Solves**

The reuse of existing factors of the previous QP solves is the most appealing way to obtain pseudo-cost estimates. In our implementation, after solving the parent NLP, we resolve the final QP to synchronize the factors with the solution of the NLP, and we then store these factors so that we can reuse them in every QP during the strong-branching phase. We use a special feature in the QP solver that allows us to hot-start the QP and is comparable to a dual-active-set method.

Table 1 shows the CPU times for some reasonably sized QP approximations. The first column gives the problem name; # ints shows the number of integer variables; and the next three columns give the CPU times for full NLP solve, single QP solve, and a hot-started QP solve, respectively.

These results show that the benefit obtained by solving just a single QP is only a factor of 2 or 3, whereas hot-started QPs are faster by a factor of up to 40.

Table 1. CPU times (s) for full NLP solve, single QP solve, and hot-started QP solve.

Problem	# Ints	Full NLP	Single QP	Hot QP
stockcycle	480	4.08	3.32	0.532
RSyn0805H	296	78.7	69.8	1.94
SLay10H	180	18.0	17.8	1.25
Syn30M03H	180	40.9	14.7	2.12

These preliminary results are encouraging because they hold the promise of a cheaper strong-branching decision for the whole tree. An alternative use of hot-started QPs that we are exploring is to replace the NLP-based tree search by a QP-based tree-search with only occasional updates to compute bounds. We believe that this approach may become competitive with the prevalent approaches to MINLP that use LP-based tree-search techniques.

4. Conclusions

We have presented new MINLP applications arising in computer science, including the optimization of IEEE 802.11 mesh networks, design of data-optical networks, optimization of energy provisioning in relay node placements for wireless sensor networks, and optimal assignment of channels to users for mesh routers in a wireless network. These models form part of a library, DIWAL; see http://wiki.mcs.anl.gov/NEOS/index.php/DIWAL.

We have investigated the tighter integration of MIP and NLP solvers for the solution of these problems. In particular, we have shown that simple heuristics for performing strong branching based on single QP information are superior to strong branching based on NLP solves.

Bibliography

[1] K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Preprint ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.

[2] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33:42–54, 2004.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Endlewood Cliffs, NJ, 1987.

[4] Randeep Bhatia, Adrian Segall, and Gil Zussman. Analysis of bandwidth allocation algorithms for wireless personal area networks. *Wireless Networks*, 12:589603, 2006.

[5] D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140, 1996.

[6] D. Bienstock and S. Mattia. Using mixed-integer programming to solve power grid blackout problems. *Discrete Optimization*, 4:115–141, 2007.

[7] P.T. Boggs and J.W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1995.

[8] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.

[9] R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE Transactions on Communications*, 25:29–47, 1977.

[10] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth. An MINLP solution method for a water network problem. In *Algorithms - ESA 2006 (14th Annual European Symposium. Zurich, Switzerland, September 2006, Proceedings)*, pages 696–707. Springer, 2006.

[11] I. Castillo, J. Westerlund, S. Emet, and T. Westerlund. Optimization of block layout deisgn problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers and Chemical Engineering*, 30:54–69, 2005.

[12] Kaikai Chi, Xiaohong Jiang, Susumu Horiguchi, and Minyi Guo. Topology design of network-coding-based multicast networks. *IEEE Transactions on Mobile Computing*, 7(4):1–14, 2008.

[13] Enrique Costa-Montenegro, Francisco J. González-Casta no, Pedro S. Rodr´guez-Hernández, and Juan C. Burguillo-Rial. Nonlinear optimization of IEEE 802.11 mesh networks. In Y. Shi et al., editor, *ICCS 2007, Part IV*, number 4490 in LNCS, pages 466–473. Springer Verlag, Berlin, 2007.

[14] R. J. Dakin. A tree search algorithm for mixed integer programming problems. *Computer Journal*, 8:250–255, 1965.

[15] V. Donde, V. Lopez, B. Lesieutre, A. Pinar, C. Yang, and J. Meza. Identification of severe multiple contingencies in electric power networks. In *Proceedings 37th North American Power Symposium*, 2005. LBNL-57994.

[16] A. M. Eliceche, S. M. Corvalán, and P. Martínez. Environmental life cycle impact as a tool for process optimisation of a utility plant. *Computers and Chemical Engineering*, 31:648–656, 2007.

[17] A. Elwalid, D. Mitra, and Qiong Wang. Distributed nonlinear integer optimization for data-optical Internetworking. *IEEE Journal on Selected Areas in Communications*, 24(8):1502–1513, 2006.

[18] R. Fletcher and S. Leyffer. Minlp (ampl input). http://www-neos.mcs.anl.gov/neos/solvers/MINCO:MINLP-AMPL.

[19] R. Fletcher and S. Leyffer. User manual for FilterSQP. Numerical Analysis Report NA/181, University of Dundee, April 1998.

[20] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–270, 2002.

[21] A. Flores-Tlacuahuac and L. T. Biegler. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Computers and Chemical Engineering*, 31:648–656, 2007.

[22] C.A. Floudas. *Nonlinear and Mixed–Integer Optimization*. Topics in Chemical Engineering. Oxford University Press, New York, 1995.

[23] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.

[24] I. E. Grossmann and Z. Kravanja. Mixed–integer nonlinear programming: A survey of algorithms and applications. In A.R. Conn L.T. Biegler, T.F. Coleman and F.N. Santosa, editors, *Large–Scale Optimization with Applications, Part II: Optimal Design and Control*, New York, 1997. Springer.

[25] Wenxuan Guo and Xinming Huang. Achieving capacity fairness for wireless mesh networks. *Wirel. Commun. Mob. Comput.*, 2009.

[26] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985.

[27] S.P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977.

[28] Y. Thomas Hou, Yi Shi, Hanif D. Sherali, and Scott F. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 4(5), 2005.

[29] N. J. Jobst, M. D. Horniman, C. A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:489–501, 2001.

[30] R. Karuppish and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers and Chemical Engineering*, 30:650–673, 2006.

[31] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[32] S. Leyffer. Generalized outer approximation. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, volume 2, pages 247–254. Kluwer Academic Publishers, 2001.

[33] Powell, M.J.D. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, 1977*, pages 144–157, Berlin, 1978. Springer–Verlag.

[34] I. Quesada and I. E. Grossmann. An LP/NLP based branch–and–bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16:937–947, 1992.

[35] A. J. Quist, R. van Gemeert, J. E. Hoogenboom, T. Ílles, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26:423–448, 1998.

[36] Waseem Sheikh and Arif Ghafoor. An optimal bandwidth allocation and data droppage scheme for differentiated services in a wireless network. ECE Technical Report 349, Purdue University, 2007.

[37] Rajnish Sinha, Aylin Yener, and Roy D. Yates. Noncoherent multiuser communications: Multistage detection and selective filtering. *EURASIP Journal on Applied Signal Processing*, 12:14151426, 2002.

[38] Majid Soleimanipour, Weihua Zhuang, and George H. Freeman. Optimal resource management in wireless multimedia wideband CDMA systems. *IEEE Transactions on Mobile Computing*, 1(2):143–160, 2002.

[39] O. V. Volkovich, V. A. Roshchin, and I. V. Sergienko. Models and methods of solution of quadratic integer programming problems. *Cybernetics*, 23:289–305, 1987.

Sven Leyffer
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Ave
Argonne, IL 60439, USA
leyffer@mcs.anl.gov