Mixed-integer nonlinear programming: Some modeling and solution issues

J. Lee

We examine various aspects of modeling and solution via mixed-integer nonlinear programming (MINLP). MINLP has much to offer as a powerful modeling paradigm. Recently, significant advances have been made in MINLP solution software. To fully realize the power of MINLP to solve complex business optimization problems, we need to develop knowledge and expertise concerning MINLP modeling and solution methods. Some of this can be drawn from conventional wisdom of mixed-integer linear programming (MILP) and nonlinear programming (NLP), but theoretical and practical issues exist that are specific to MINLP. This paper discusses some of these, concentrating on an aspect of a classical facility location problem that is well-known in the MILP literature, although here we consider a nonlinear objective function.

Introduction

In a simplified view, business optimization involves important aspects such as the identification of one or more high-value opportunities, data collection and analysis, modeling, mathematical optimization, and solution delivery. None of these aspects can be fully considered in isolation, because high-value applications benefit from feedback among these exercises. Nevertheless, we focus on the part of the larger challenge that involves modeling and mathematical optimization.

Although many high-value opportunities exist for which optimization-based solutions could in principle be applied, a significant inhibiting factor for widespread use is the custom nature of much of the modeling and optimization. This customization arises primarily from a mismatch between natural optimization models and available optimization solvers. This mismatch is largely due to the inherent complexity of applications and the limited scope of efficient algorithms. This paper focuses on recent efforts to ameliorate this situation through the use of mixed-integer nonlinear programming (MINLP).

As general background to various optimization approaches, linear programming is concerned with methods for optimizing a linear function of many variables, subject to bounds on a finite set of additional linear functions. Nonlinear programming is associated with optimization in instances in which some of the relevant functions are nonlinear. The term *mixed-integer* indicates that some of the variables can take only discrete values, while others are continuous. Thus, the term *mixed-integer nonlinear programming* refers to mathematical programming with continuous and discrete variables and nonlinearities in the objective function and constraints. Practical applications of MINLP include portfolio optimization, design of water distribution networks, design of complex distillation systems, and optimization of manufacturing and transportation logistics.

In the following, $\mathbb{R}$ denotes the real numbers and $\mathbb{Z}$ denotes the integers. The form of our MINLP problem is

$$P: \quad \begin{array}{l} \text{minimize } f(x) \\ \text{subject to } g(x) \leq b \\ x \in S, \end{array}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and the $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ are twice continuously differentiable functions, b is a real m-vector, and $S := \mathbb{R}^{n-k} \times \mathbb{Z}^k$ (that is, the first $n - k$ variables are continuous, and the last k variables are integers). This is a fairly general setting, accommodating both continuous and discrete variables as well as inequality constraints involving smooth functions. At two extremes, P reduces to a standard smooth nonlinear programming (NLP) model when $k = 0$, and to a standard mixed-integer linear programming (MILP) model when the functions f and g

0018-8646/07/$5.00 © 2007 IBM

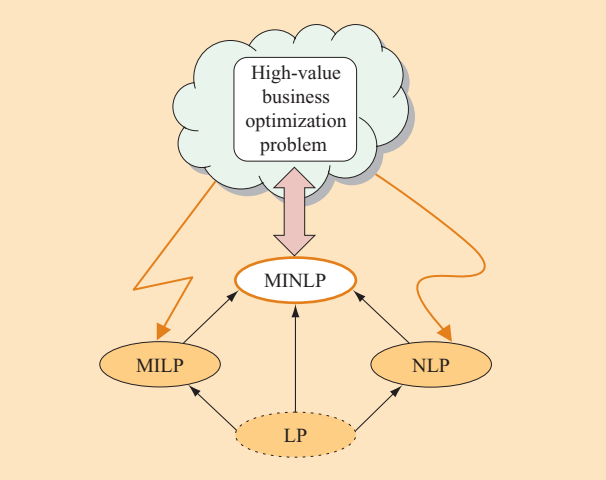

Figure 1

Business optimization and a solver hierarchy.

are linear. When we have both of these extremes at the same time, we have a standard linear programming (LP) model. **Figure 1** is a high-level encapsulation of many of the concepts in this Introduction and highlights the relationships between various model and solver types. The figure is discussed in more detail toward the end of the Introduction.

In recent years, there have been enormous advances in solver technology for LP, MILP, and NLP. Commercial tools such as ILOG CPLEX** [1] for LP and MILP, and SNOPT** [2] for NLP are quite robust for problems that even a decade ago were not tractable. Open-source tools such as COIN-OR's Clp for LP, Cbc for MILP, and Ipopt [3] for NLP are attractive alternatives in many situations. From a user's perspective, the customizability and lack of cost make the open-source tools an attractive alternative to the commercial tools. All of these solvers are accessible by modeling languages such as AMPL** [4] and GAMS [5] and can also be called from languages suitable for high-performance computing such as C/C++.

Many high-value applications are *naturally* modeled as MINLP problems. Aspects exist that are inherently nonlinear (e.g., engineering aspects, such as pressure loss due to friction), as well as other aspects that are inherently discrete and of exponential complexity (e.g., in problems concerning simultaneous choices from many small sets of alternatives).

Given such problems, what is the modeler to do when confronted with a high-value application that is naturally modeled as an MINLP? Aside from reformulating the problem, the traditional approaches involve either relaxing the integrality restrictions and using an NLP solver, or judiciously replacing the nonlinear functions with piecewise-linear ones. In the former case, it may be difficult to use the solution of the NLP to find a good solution satisfying the integrality restrictions. In the latter case, the cost of this approximation is paid by an increase in the number of discrete variables, which are used to manage the linear pieces. The running time of MILP solvers tends to increase, often dramatically, as the number of discrete variables increases. Thus, this approach can drastically limit the size of problems that we can solve. If we artificially limit the number of pieces in the approximation, we lose accuracy in the approximation. Therefore, whether we use an NLP or MILP approach, we must deal with solution methods that may not yield even feasible solutions within reasonable time limits.

Of course we can make some progress by using MILP solvers. For example, in the case of piecewise-linear approximation, fairly simple methods exist for separable functions [i.e., $f : \mathbb{R}^n \mapsto \mathbb{R}$ is separable if it has the form $f(x) = \sum_{j=1}^{n} f_j(x_j)$]. Here we can use some well-known specialized techniques (so-called "SOS Type-2" methods; see [6]) to handle these situations. Other specialized techniques have been developed that are applicable in situations involving partially separable functions, in which a limited number of variables appear in summands of the functions (see [7] and also [8], which develop related techniques and apply them to gas-network optimization problems). However, experience has shown that these models have limited value for large instances and those requiring accurate solutions. Moreover, the successful application of these methods is currently far from automatic.

On the other hand, MILP approaches to MINLP problems have certain attractive aspects. With a sufficiently fine piecewise-linear approximation, most MILP methods work with good approximations to lower and upper bounds on the optimum value and seek to shrink the gap between the two. Moreover, the methods are designed so that after a finite number of steps, which may be quite large, an optimal solution to the approximation is obtained and verified.

MILP is a very powerful paradigm, and the value of MINLP is not limited to attacking piecewise-linear approximations. For example, specialized heuristic approaches based on MILP have been developed for bilinear-programming models that are used for obtaining good solutions to certain cutting-stock problems (see [9]). The cutting-stock problem occurs in various industries, such as in a paper mill where stock rolls of paper must be cut to satisfy demand for rolls of various smaller widths and in such a way as to minimize trim loss.

An NLP approach to MINLP problems has the virtue of working with the correct nonlinear functions, but the discrete nature of some of the variables is ignored. In

many applications, no simple method exists to recover a good solution to the underlying MINLP from a solution of the NLP relaxation, and so an MINLP method must be developed that will find a way to impose the discrete restrictions. (The term *NLP relaxation* refers to an NLP that arises as a result of allowing the integer variables of the MINLP to assume continuous values.)

Another drawback of an NLP approach is that usually the NLP relaxation does not have a convex feasible region. (The feasible region of an optimization model is the set of solutions that satisfies its constraints.) Since most NLP methods seek a local optimum, it is virtually impossible to verify the global quality of a solution to a large and/or complex model.

In Figure 1, the vertical (piecewise-linear) zigzag arrow pointing downward to MILP alludes to the modeling compromises that MILP might demand—typified by piecewise linearization. Similarly, the (curvilinear) arrow pointing downward to NLP represents the modeling compromises that NLP might demand—typified by rounding variables that should be discrete. The arrows pointing upward indicate the containment relationship between the various paradigms and indicate how solution technology from the more basic paradigms can be built upon to develop a solution technology for MINLP—which is the paradigm that requires the fewest modeling compromises among the ones that we consider.

We have used much of the algorithmic wisdom gleaned from research in MILP and NLP, and harnessed open-source solvers for these two paradigms, in order to create a new open-source solver, BONMIN (Basic Open-source Nonlinear Mixed INteger programming), for MINLP (see [10] and [11]) which is distributed under the Common Public License (CPL) at COIN-OR (*www.coin-or.org*). We have already observed some success on, for example, difficult water-network optimization problems [12], as well as portfolio optimization problems. Aside from further developing BONMIN, we must now learn more about MINLP modeling and how to tune codes such as BONMIN to perform well on various MINLP models.

Some conventional wisdom for MILP

The theory and practice of MILP is a well-studied area [13]. Nonetheless, fundamental results of computational complexity delineate limitations on the potential for success. The concept of tight formulation is extremely important. We discuss this in some detail in the context of a particular model, the *uncapacitated facility location (UFL) problem*. The UFL problem involves satisfying the demand of many customers for a single commodity by opening production facilities from a finite set of potential locations. Costs are associated with opening facilities, and per-unit shipping costs are associated with moving the commodity from facilities to customers. These costs typically depend on the facility–customer pair involved. The problem is uncapacitated in the sense that there is no upper limit on the production capacity of any facility. Next, we model this problem as an MILP. We focus on the UFL problem because it is rather well understood from the MILP viewpoint, but we observe that not all of this MILP knowledge transfers to a very similar nonlinear version of the UFL problem.

We have the discrete variables $y_i :=$ indicator variable for facility i and continuous variables $x_{ij} :=$ proportion of customer j demand satisfied by facility i. Because we are working with proportions, we obviously require that

$$x_{ij} \geq 0 \quad \text{for } i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n.$$

Here we have potential locations for facilities numbered $i = 1, 2, \ldots, m$ and customers numbered $j = 1, 2, \ldots, n$. Moreover, we require satisfaction of demand:

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \text{for } j = 1, 2, \ldots, n.$$

Implicit in the definition of the indicator variables y_i is that they are constrained to be integers from the interval [0, 1]. In other words, the restriction that y_i is binary is trivially modeled in the context of MILP.

Finally, to enforce that we ship only commodities from open facilities, we can use the *weak forcing constraints*:

$$\sum_{j=1}^{n} x_{ij} \leq n y_i \quad \text{for } i = 1, 2, \ldots, m.$$

In the simplest case, we seek to minimize the linear cost function

$$\sum_{i=1}^{m} c_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$$

specified by the following:

- Facility cost data: $c_i :=$ cost of operating facility i.
- Commodity-transportation cost data: $d_{ij} :=$ cost of satisfying all of the demand of customer j from facility i.

An alternative to the set of weak forcing constraints is the set of *strong forcing constraints*:

$$x_{ij} \leq y_i \quad \text{for} \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n.$$

Although the weak and strong forcing constraints are equivalent on $S := \{(y, x) \in \{0, 1\}^m \times [0, 1]^{mn}\}$, they are not equivalent on the relaxed domain $S_R := \{(y, x) \in [0, 1]^m \times [0, 1]^{mn}\}$, that is, with the integrality restrictions on y_i relaxed. It is easy to see

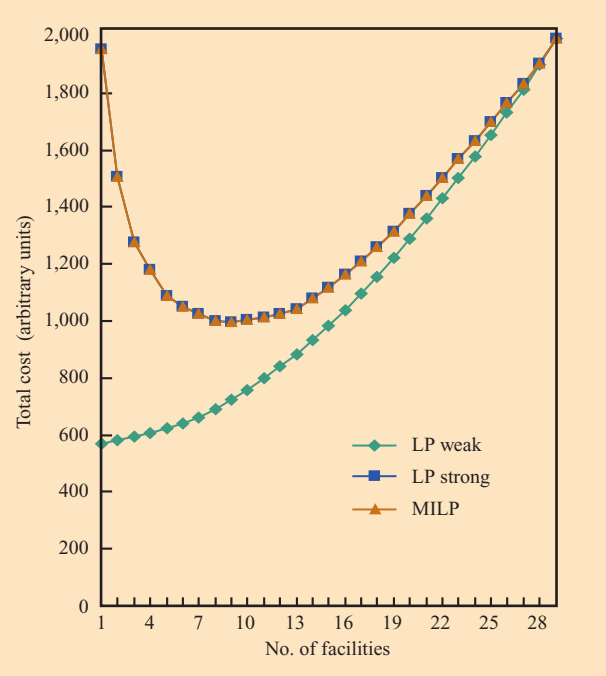

Figure 2

Quality of LP formulations of a linear-objective uncapacitated facility location problem.

that on S_R the strong constraints imply the weak ones, since the weak constraints arise by summing the strong ones over $j = 1, 2, \ldots, n$. On the other hand, for example when $m = n$, we can set $x_{ii} = 1$ for $i = 1, 2, \ldots, m$ and 0 otherwise, and set $y_i = (1/n)$, and we see that the weak constraints are all satisfied (at equality), but the strong ones are violated (by quite a lot) for all i, j such that $i = j$.

Conventional MILP wisdom holds that strong formulations are better for MILP solvers that use LP relaxations (and these are the type that dominate the MILP-solver space). Indeed, this is demonstrated with simple computational experiments. **Table 1** shows some indicative results for a randomly generated linear-objective UFL instance having $m = 30$ and $n = 100$. Our solver is `Cbc`, which we accessed via `AMPL` (a modeling language for mathematical programming), and experiments were performed using a notebook computer running Windows** XP. The column labeled *Nodes* provides information on the number of search-tree nodes considered in order to find and prove global optimality. The column labeled *Time* indicates the number of seconds of computation. The weak and strong formulations use the weak and strong forcing constraints, respectively, and we observe the expected result that the strong formulation is vastly superior. The last row of the table demonstrates that advanced algorithmics, in this case

Table 1 UFL MILP results.

	Nodes	*Time* (s)
Weak formulation	10,616	332.24
Strong formulation	0	0.17
Strong algorithmic	2	1.69

constraint tightening and generation, can sometimes achieve what a knowledgeable modeler can accomplish. For that run, we used the *weak* formulation, but we turned on the "integer preprocessing" and "cut-generation" procedures of `Cbc`.

We can make a more detailed comparison of the weak and strong forcing constraints. Because the transportation cost is linear in the x_{ij}, once the facility variables are fixed, each customer's demand is fully assigned to the "closest" open facility (that is, for each customer j, all of its demand is satisfied from a facility i that minimizes d_{ij}).

The weak relaxation is poor, since the y_i variables can "cheat" the weak forcing constraints in the LP and assume very small nonzero values. This does not happen to the same extent with the strong forcing constraints. (The term *cheat* suggests that the y_i variables, by assuming continuous values, can satisfy the weak constraints that are meant to enforce logical restrictions.)

Figure 2 compares the LP minimum and the MILP minimum as the number of facilities is fixed at each possible value from one up through m. Clearly the strong formulation is quite effective, and the weak one is quite poor, except when the number of facilities is required to be high. There are several noteworthy points:

- The LP bound based on the strong forcing constraints is significantly better than the LP bound based on the weak forcing constraints. Hence, it is recommended that practitioners work with the strong formulation or use a modern MILP solver that has constraint tightening and generation. We emphasize, however, that modern MILP solvers do not in general obviate the need for strong MILP formulations.
- The weak relaxation predicts that the overall optimal MINLP solution will involve only one facility. This is in stark contrast to the fact that the overall MILP optimum makes use of eight facilities.
- The weak relaxation predicts that the optimal MILP has a solution cost that increases consistently with the number of facilities. However, the MILP cost consistently *decreases* as the number of facilities is increased from one through nine.

- The quality of the weak relaxation, as a lower bound, is actually quite good for larger numbers of facilities. For example, the MILP optimal value is within five percent of the LP optimal value once the number of facilities reaches twenty-two.

Many other issues are involved in the successful modeling and solution of MILP problems. We do not discuss them in any detail, but we list some of the key issues together with representative pointers to relevant literature:

- Preprocessing and coefficient improvement; for example, see [14].
- Implicit formulations (i.e., generating constraints and variables); see [15] for use in a business application.
- Underlying LP methods and performance (including such topics as interior point algorithms vs. simplex methods; primal vs. dual; taking advantage of sparsity; and warm-starting successive LPs); see [15].
- Difficulty of exploiting massively parallel architectures; for example, see [16].
- Branching rules and strong branching; for example, see [15, 17].
- Heuristics; for example, [18].

Some conventional wisdom for NLP

In NLP, we have the benefit of being able to accurately model various nonlinear phenomena. This makes the paradigm particularly attractive for applications with complex engineering aspects. We quickly find that many of the important applications do not lend themselves to convex formulations, so we are able to guarantee only local optimality. Nevertheless, significant value is possible in such local optimization, and we can make more progress than by using a completely naïve approach if we employ multiple starting points as well as methods that may allow uphill steps.

Effective modeling pays attention to the need to provide function evaluations and usually first derivatives to solvers. Second derivatives should exist and should probably be smooth, but the user is not normally required to provide this. Sparsity has a key effect on performance. Many different algorithmic choices can be made, primarily interior point vs. active set, and many choices exist for the underlying unconstrained NLP solver and even for solving the basic linear systems that arise. Scaling and moderating the degree of nonlinearity are essential for numerical stability. Sometimes judicious variable transformations can have an enormous impact on the quality of solutions. An older reference which still bears reading is Chapter 7 of [19].

Since MILP methods rely largely on LP relaxation, such methods mostly inherit the robustness, i.e., the stable numerical properties, of LP methods. However, caution is required because the use of cutting planes may introduce some numerical instability. NLP methods are generally not nearly so robust as MILP methods. In fact, some of this unstable behavior is inherent in particular NLP methods. For example, so-called sequential quadratic programming (SQP) methods of NLP solve a sequence of linearized problems. These linearizations may be infeasible, even when the original problem is feasible.

Toward the development of wisdom for MINLP

First, we prominently note that because MILP and NLP are special cases of MINLP, we need to be aware of the theoretical and practical knowledge associated with these subdisciplines. However, we should seek to become more knowledgeable if we hope to take full advantage of the broader domain.

Associated with an MINLP model is its continuous relaxation, obtained by relaxing S to $S_R := \mathbb{R}^{n-k} \times \mathbb{R}^k$. Thus, the continuous relaxation of an MINLP model is an NLP model.

At one extreme, when f is concave and g is convex, local optima are found as extreme points of the feasible region of the relaxation. If we are sufficiently fortunate that such a point is feasible for the underlying MINLP, we have found a globally optimal solution to the MINLP. If such a point is not feasible for the MINLP, we may try to tighten the relaxation (by appending inequalities designed to cut off such a point, but preserving the feasible region of the underlying MINLP). Unfortunately, it is not possible to reliably find a global minimum of the NLP relaxation in this case, because a local optimum need not be global.

At another important extreme, when the functions f and g are all convex, a local optimum of the NLP model is also a global optimum, and thus NLP algorithms that seek local optima (all efficient NLP algorithms) in fact find global optima. Unfortunately, local optima may be on the interior of the feasible region, and these points are likely to be infeasible for the underlying MINLP model. Moreover, the NLP solution may also be quite far from MILP optima. In addition, since an optimum may be within the convex hull of the feasible solutions of the MINLP, it is not possible to usefully tighten the MINLP formulation. All of this is illustrated in the two-dimensional example of **Figure 3**, where we see contours of a convex objective function that we seek to optimize on the lattice points of the shaded polygon. We see the NLP optimum at the red point, while MINLP optima, circled in blue, are at points that cannot be obtained by rounding. At this stage, we must rely on some disjunctive

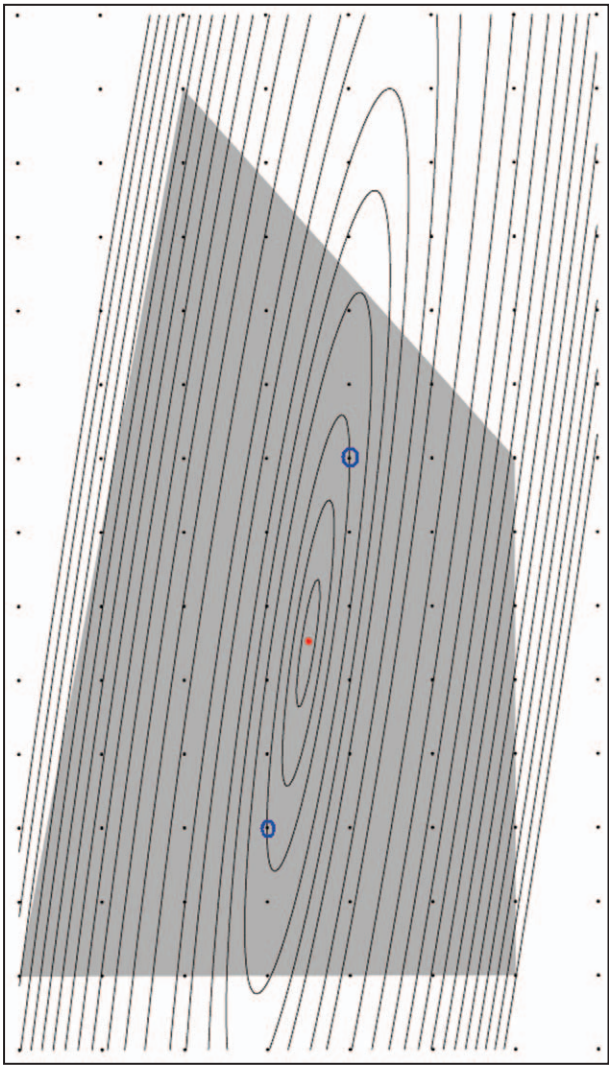

Figure 3

Geometry of a two-dimensional convex objective MINLP function. The axes represent two decision variables, x_1 and x_2.

algorithmic reasoning (i.e., branching, or subdividing the feasible region).

The introduction of [20] discusses at some length a small example focusing on aspects of the convexity and concavity issue that we have just discussed.

Combining the previous two cases, we see that linear objectives and convex constraints are particularly attractive for MINLP. Linear constraints are even more attractive from a computational standpoint, but then we are in the realm of MILP.

Considerable literature exists on MINLP. Some of this literature is focused on the case in which the continuous relaxation is a convex optimization problem. For this case, algorithmic approaches that we may use include. NLP-based branch-and-bound [21], the outer-approximation decomposition algorithm [22], the branch-and-cut algorithm of Quesada and Grossmann [23], and a hybrid outer-approximation-based branch-and-cut algorithm [10]. All four of these methods are implemented in BONMIN.

For the nonconvex case, conventional approaches include various heuristic methodologies as well as global optimization techniques. A central approach of global optimization is "spatial branch-and-bound," in which the feasible region is repeatedly subdivided and the objective function is bounded below, for each subproblem, by solving a convex relaxation. The success of the approach depends on the method being able to branch in such a way as to improve the bounds. Some entry points to this literature are [24–27]. Rather than following the approach of global optimization, the NLP-based branch-and-bound algorithmic option in BONMIN includes some simple techniques to make the search robust in the presence of nonconvexity, with the goal of finding a feasible solution that only approximately minimizes the objective function.

MINLP and the UFL

It is interesting to compare an optimization model with its continuous relaxation. Lee and Morris [28], using a "volumetric" view, did this analytically for various polytopes. They were concerned primarily with linear constraints, but the motivation for using volume as a measure of the quality of a model (relative to its continuous relaxation) relates to nonlinear objective functions. With this viewpoint, Lee and Morris analytically compared the weak and strong forcing constraints of the UFL problem and demonstrated that when m grows slowly with n, the weak formulation is not much weaker than the strong formulation.

As we have already seen, in the context of MILP, the constraints of the UFL are particularly well understood. Besides the favorable properties of the strong forcing constraints, it is clear that once the facility variables are fixed, the customers are just assigned to the "closest" open facility. We have performed some experiments with instances of the UFL problem having a separable convex quadratic objective function:

$$\sum_{i=1}^{m} c_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}^2 .$$

From the NLP point of view, this is an extremely useful objective function. However, from the point of view of MINLP, this particular objective function is disastrous. Convexity here means dis-economies of scale. It is advantageous to split each customer's demand over several facilities. Since transportation-cost minimization encourages the x_{ij} variables to assume *very* small fractional

values, the y_i variables "cheat" the forcing constraints, weak or strong, in the NLP and assume small nonzero values as well. **Figure 4** compares the NLP minimum and the MINLP minimum as the number of facilities is fixed at each possible value from one up through m.

We can make several observations regarding Figure 4:

- The NLP bound is not significantly better using the strong forcing constraints as compared with the weak forcing constraints. Hence, it may be preferable to work with the weak formulation because the NLPs solve faster.
- Both NLP relaxations predict that the overall optimal MINLP solution will involve only one facility. This is in stark contrast to the fact that the overall MINLP optimum uses eight facilities.
- Both NLP relaxations predict that the optimal MINLP solution cost increases consistently with the number of facilities, but the MINLP cost consistently *decreases* as the number of facilities is increased from one up through eight.
- The quality of both NLP relaxations (as a lower bound) is actually quite good for larger numbers of facilities. For example, the MINLP optimal value is within five percent of the NLP optimal value once the number of facilities reaches eighteen.

We have used `BONMIN` to solve this instance to optimality, using both weak and strong formulations. Note that a good branching rule was essential for obtaining reasonable performance; branching with priority given to facilities with higher costs is an effective technique. Our results are summarized in **Table 2**. A few points are clear when Tables 1 and 2 are compared:

- Even for the weak formulation, the time required to solve each NLP subproblem is much greater than for the corresponding LP subproblem.
- The number of nodes for either formulation is significantly greater for the MINLP than for the corresponding MILP.

None of this is surprising. The more interesting comparison is between the weak and strong formulations of the MINLP. In particular, the number of nodes for the weak formulation of the MINLP is only modestly worse than the number of nodes for the strong formulation (very different from the MILP situation), but the running time for the weak formulation is significantly lower for the weak formulation (also very different from the MILP situation).

Of course, it may be practical to consider heuristics when solving business optimization problems. Borrowing an idea from [9], we suggest a family of heuristics based on solving smaller MINLP problems. The idea is also closely related to [29].

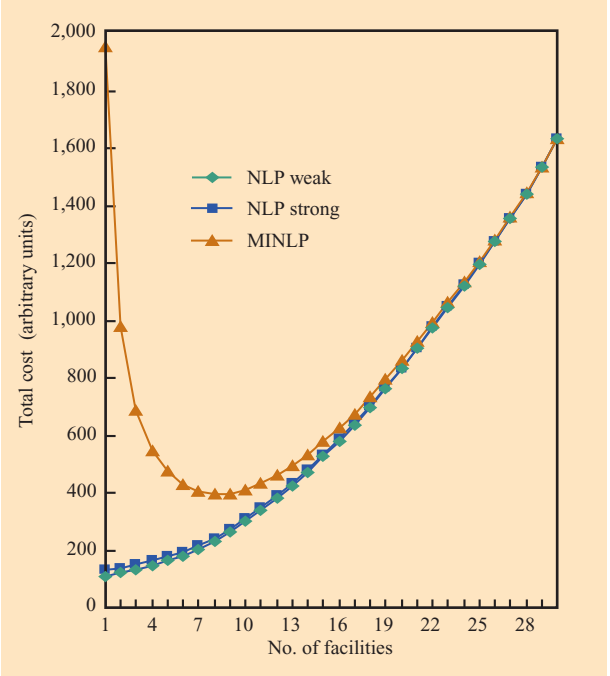

Figure 4

Quality of NLP formulations of a quadratic-objective uncapacitated facility location problem.

Table 2 UFL MINLP results.

	Nodes	*Time* (s)
Weak formulation	45,901	16,697.46
Strong formulation	29,277	21,206.56

Let S be a subset of the set of facilities $\{1, 2, \ldots, m\}$. Initially, we may take $S := \emptyset$, or $\{1, 2, \ldots, m\}$, or perhaps a known set of facilities yielding a feasible solution. We choose a positive integer parameter k, which serves as a Hamming radius around S. We append the following inequalities to our MINLP:

$$\sum_{i \in S} (1 - y_i) + \sum_{i \notin S} y_i \leq k, \tag{1}$$

$$|S| - 1 \leq \sum_{i=1}^{m} y_i \leq |S| + 1, \tag{2}$$

$$\sum_{i \notin S} y_i \leq \left\lfloor \frac{k+1}{2} \right\rfloor, \tag{3}$$

$$\sum_{i \in S} y_i \geq |S| - \left\lceil \frac{k+1}{2} \right\rceil . \quad (4)$$

Here, the $\lfloor x \rfloor$ symbol denotes the greatest integer less than or equal to x, and $\lceil x \rceil$ denotes the least integer greater than or equal to x. The inequality (1) forces the y_i to select a set of facilities that is not very different from S. The inequality (2) further restricts the y_i so that the number of facilities chosen is within one of the number chosen by S. The remaining inequalities (3, 4) are implied by (1, 2) and the binary nature of the y_i, and they strengthen the NLP relaxation when k is even.

By repeatedly solving the MINLP, letting $S := \{i : y_i = 1\}$ with respect to the y_i from the previous stage, we simply iterate until no improvement is found. If we set $k := m$, we are just solving the full MINLP in one stage. Thus, we suggest setting the value of k substantially smaller in order to limit the computational effort expended at each stage. Even with $k = 1$, this heuristic found the MINLP minimum on our UFL data set. An additional feature of the heuristic is that it is completely generic for problems in which the integer variables are constrained to be 0/1-valued; in fact, in principle, it is easy to see how to adapt this to general integer variables.

Conclusions and future directions

Tools currently being developed for MINLP problems have tremendous potential for business optimization problems. Realizing this potential will require unique modeling wisdom for this domain in combination with advances in MINLP solver technology.

Considerable work has to be done before MINLP solvers will have the impact that MILP tools have had for business optimization problems. We can highlight a few areas of current investigation that should have considerable impact on our ability to solve difficult problems:

- Development of general-purpose and specialized heuristics (see for example [9] and [30]).
- Development of effective branching rules.
- Development of effective "warm-starting" techniques for rapidly solving children NLPs from the solution of a parent NLP.
- Effective use of parallel computing.
- Development of algebraic approaches, generalizing ideas such as those of [31].

Future versions of BONMIN should incorporate many of these features and thus enable us to expand the applicability of the MINLP paradigm.

**Trademark, service mark, or registered trademark of ILOG, Inc., Stanford University, UC San Diego, AMPL Optimization LLC, or Microsoft Corporation in the United States, other countries, or both.

References

1. ILOG-Cplex 9.0 User's Manual, 2003; see *http://www.ilog.com/*.
2. P. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.* **47**, No. 1, 99–131 (2005).
3. A. Waechter and L. T. Biegler, "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Math. Program.* **106**, No. 1, 25–57 (2006).
4. R. Fourer, D. Gay, and B. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole, Stamford, CT, 2003.
5. A. Brooke, D. Kendrick, A. Meeraus, and R. Raman, *GAMS: A User's Guide*; see *http://www.gams.com/docs/document.htm*.
6. E. M. L. Beale and J. J. H. Forrest, "Global Optimization Using Special Ordered Sets," *Math. Program.* **10**, 52–69 (1976).
7. J. Lee and D. Wilson, "Polyhedral Methods for Piecewise-Linear Functions I: The Lambda Method," *Discrete Appl. Math.* **108**, No. 3, 269–285 (2001).
8. A. Martin, M. Möller, and S. Moritz, "Approximation of Non-Linear Functions in Mixed Integer Programming," presented at the workshop on Integer Programming and Continuous Optimization, Chemnitz, Germany, November 7–9, 2004.
9. J. Lee, "In Situ Column Generation for a Cutting-Stock Problem," *Computers & Oper. Res.* **34**, No. 8, 2345–2358 (2007); see *www.sciencedirect.com*.
10. P. Bonami, A. Waechter, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, et al., "An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs," *Research Report RC-23771*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 2005.
11. P. Bonami and J. Lee, BONMIN Users' Manual; see *http://projects.coin-or.org/Bonmin*.
12. C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth, "An MINLP Solution Method for a Water Network Problem," *Algorithms—ESA 2006*, Y. Azar and T. Erlebach, Editors, *Proceedings of the 14th Annual European Symposium*, Zurich, Switzerland, Springer, New York, 2006, pp. 696–707.
13. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, Hackensack, NJ, 1988.
14. M. W. P. Savelsbergh, "Preprocessing and Probing for Mixed Integer Programming Problems," *ORSA J. Computing* **6**, No. 4, 445–454 (1994).
15. R. Anbil, R. Tanga, and E. L. Johnson, "A Global Approach to Crew-Pairing Optimization," *IBM Syst. J.* **31**, No. 1, 71–78 (1992).
16. Q. Chen, M. Ferris, and J. T. Linderoth, "FATCOP 2.0: Advanced Features in an Opportunistic Mixed Integer Programming Solver," *Ann. Oper. Res.* **103**, 17–32 (2001).
17. J. T. Linderoth and M. W. P. Savelsbergh, "A Computational Study of Branch and Bound Search Strategies for Mixed Integer Programming," *INFORMS J. Computing* **11**, No. 2, 173–187 (1999).
18. L. Bertacco, M. Fischetti, and A. Lodi, "A Feasibility Pump Heuristic for General Mixed-Integer Problems," *Discrete Optimization* **4**, 63–76 (2007).
19. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York, 1981.
20. J. Lee, *A First Course in Combinatorial Optimization*, Cambridge University Press, New York, 2004.
21. O. K. Gupta and V. Ravindran, "Branch and Bound Experiments in Convex Nonlinear Integer Programming," *Manage. Sci.* **31**, No. 12, 1533–1546 (1985).

22. M. Duran and I. E. Grossmann, "An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs," *Math. Program.* **36**, No. 3, 307–339 (1986).
23. I. Quesada and I. E. Grossmann, "An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems," *Computers & Chem. Eng.* **16**, No. 10/11, 937–947 (1992).
24. L. Liberti and N. Maculan, *Global Optimization: From Theory to Implementation*, Chapter: "Nonconvex Optimization and Its Applications," Springer, New York, 2006.
25. C. Audet, P. Hansen, and G. Savard, *Essays and Surveys in Global Optimization*, GERAD 25th Anniversary Series, Springer, New York, 2005.
26. M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, Kluwer Academic Publishing, Dordrecht, The Netherlands, 2002.
27. C. A. Floudas, *Deterministic Global Optimization*, Kluwer Academic Publishing, Dordrecht, The Netherlands, 2000.
28. J. Lee and W. D. Morris, Jr., "Geometric Comparison of Combinatorial Polytopes," *Discrete Appl. Math.* **55**, No. 2, 163–182 (1994).
29. M. Fischetti and A. Lodi, "Local Branching," *Math. Program.* **98**, 23–47 (2003).
30. P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot, "A Feasibility Pump for Mixed Integer Nonlinear Programs," *Research Report RC-23862*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 2006.
31. J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, and R. Yoshida, "A Computational Study of Integer Programming Algorithms Based on Barvinok's Rational Functions," *Discrete Optimization* **2**, No. 2, 135–144 (2005).

Received April 6, 2006; accepted for publication September 22, 2006; Internet publication May 23, 2007

Jon Lee *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (jonlee@us.ibm.com)*. Dr. Lee manages the Discrete Optimization group, which is in the Optimization Center of the Mathematical Sciences Department. He received his B.S. (1981), M.S. (1984), and Ph.D. (1986) degrees from Cornell University. From 1985 to 1993, he was on the faculty of Yale University, and from 1993 to 2000 on the faculty of the University of Kentucky. Dr. Lee was a research visitor at the Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium, for the academic years 1991–1992 and 1999–2000, and an Adjunct Professor in the Industrial Engineering and Operations Research Department at Columbia University in 2003. He has worked at IBM since 2000. Dr. Lee is the author of *A First Course in Combinatorial Optimization* (Cambridge University Press) and was founding Managing Editor of the journal *Discrete Optimization*. He is currently Associate Editor of the journal *Discrete Applied Mathematics*, an Adjunct Professor at the Leonard N. Stern School of Business, New York University (since 2002), a Permanent Member of DIMACS, a Full Member of the COIN-OR Foundation, and an external member of the Computational Optimization Research Center (CORC) of Columbia University.