MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming

Michael R. Bussieck, GAMS Development Corp, 1217 Potomac Street, NW, Washington, D.C. 20007, USA
Email: MBussieck@gams.com
Arne Stolbjerg Drud, ARKI Consulting & Development A/S, Bagsvaerdvej 246A, 2880 Bagsvaerd, Denmark
Email: ADrud@arki.dk
Alexander Meeraus, GAMS Development Corp, 1217 Potomac Street, NW, Washington, D.C. 20007, USA
Email: AMeeraus@gams.com

Abstract:

The paper describes a new computerized collection of test models for mixed-integer nonlinear programming. Since there is no standard format for nonlinear models, the model collection is augmented with a translation server that can transform the models from their basic GAMS format into a number of other formats such as AMPL, BARON, LINGO, and MINOPT. The translation server can also be used to transform industrial models with confidential information in a way that allows many of these models to be distributed to the research community as highly relevant algorithmic test models.

Introduction:

The continuing interaction between research and applications is one of the fascinating aspects of practical Mathematical Programming. New solvers are being developed and tested on practical models from many application areas. For academic researchers, access to a wide set of interesting models with different characteristics has always been important, but also a bottleneck for their work. When David Gay in 1985 started the Netlib collection of Linear Programming (LP) models in the industry standard MPS format, [12], he created an institution that has had an important impact on the field of Linear Programming. Today, all papers describing new developments in LP will in some form refer to experiments with some of these models, and an LP code is not consider acceptable unless it performs reasonably well on the Netlib collection. The MIPLIB collection developed later by Bixby et. al. [1] has filled a similar need in the field of Mixed Integer Linear Programming (MIP).

In the rapidly growing field of Mixed Integer Nonlinear Programming (MINLP) we do not have a similar computerized and easily available collection of models. There are some books with test models for global optimization, e.g. [7] and [8], and some of these models are MINLP models. Floudas and Leyffer have web sites with models used to test their own codes, [9] and [13], some written in GAMS and others in MINOPT and AMPL. The main reason we do not have a common computerized collection of models is that we do not even have a common format like the MPS format for representing the underlying nonlinear models. Many attempts have been made over the years to create a standard format for Nonlinear Programming (NLP), e.g. the SIF format [4] used in the CUTE collection [5], but none of them have been widely accepted. The type of information required by different NLP algorithms is simply too different, see also [2]. Practical NLP and MINLP models are therefore today often represented in modeling system languages, such as GAMS [3] or AMPL [10], that translate the models into the format required by an algorithm.

When creating a collection of test models it is important to have a wide selection of models including models with practical relevance from many different application areas. It is usually not difficult to get access to models developed by university researchers, but practical models developed in commercial organizations will often be based on confidential data or proprietary information. Although they often have a different structure than university models and therefore can be very useful for algorithm developers, we seldom have access to the commercial models.

In this paper we describe a new collection of MINLP test models that is available through a new web site called "MINLP World" at http://www.gamsworld.org/minlp. We have attacked both problems of format and confidentiality by using a new model translator built into the GAMS system. This translator and its output will be described in section 1. Information about the MINLP library is given in section 2. Other information about the MINLP initiative and a conclusion is given in section 3.

1. CONVERT - A GAMS Model Translator

Most practical models involving nonlinearities are today developed in a modeling language, and most practical MINLP solvers are linked to a modeling system. It is therefore natural to base a collection of MINLP test models on a widely used modeling system. We have selected the GAMS modeling system. Since some MINLP solvers do not have an interface to GAMS, the model library will only be of general use if we also can offer test models for these MINLP solvers. Building GAMS interfaces to all potential solvers would be an intractable task because some solvers are tightly

connected to a particular modeling system. Instead, we have selected to build a translator that can transform a GAMS model into many other formats. To use the MINLP model library for testing, the developer of a solver must build an interface to at least one of these formats.

Modeling languages have a rich syntax that is usually based on sets and indexed variables, equations, and parameters. This syntax and the corresponding structure in the model and the data is very useful for the model developer. However, with very few exceptions, e.g. Fragniere et.al. [11], the structure is not used by the solvers. Most solvers look at the world as consisting of a linear list of variables, X_1 to X_n, a linear list of equations or constraints, E_1 to E_m, plus the relationship between these variables and equations represented in some form. It is therefore acceptable that a translator removes the structure as long as the model as seen by the solver remains unchanged.

The GAMS translator transforms models into a very simple internal scalar format. This internal format can then be written out in many different formats. With GAMS as output format, the scalar model consists of

- declarations of the variables with extra declarations for the subsets of positive, integer, or binary variables,
- declarations of the equations named E1 to EM,
- the symbolic form of these equations, and
- assignment statements for non default bounds and initial values.

All operations involving sets are unrolled, and all expressions involving parameters are evaluated and replaced by their numerical values.

Since there are no sets or indexed parameters in the scalar models, most of the differences between modeling systems have disappeared. Therefore, the GAMS format can be easily transformed into the format of another language. In AMPL, the keyword "`var`" is used instead of "`Variable`", bounds and initial values are written using a different format, the equation declarations are missing, the equation definitions start with "`subject to Ei`" instead of "`Ei..`", and a few operators are named differently. In a few cases a model cannot be translated into a particular language because special functions (e.g. `errorf`) or variable types (e.g. `SemiCont`) are not available in that language. Figure 1 represents a user-written GAMS model for trim loss minimization, figures 2 and 3 represent its scalar GAMS and AMPL versions.

The proprietary parts of an industrial model are usually hidden in the names of sets, parameters, variables, and equations; the names of set elements, the numerical values of parameters, the structure of the symbolic equations, and the relationships between these items. During the translation of a user-written GAMS model into scalar format, most of this information is lost. All that is left is the size of the model, the structure of the Jacobian, and some linear and nonlinear relationships with numerical coefficients. The developer of a model will often be unable to recognize his own model. Industrial users with an interest in improving the solvers they use themselves can therefore in many cases be allowed to release the scalar version of their model to a model library like MINLPLib. Many of the initial models in the library are indeed of industrial origin and we expect that the number of these models will grow considerably in the future.

The GAMS translator is implemented as a regular GAMS solver called *convert*. Convert is part of any GAMS system and, besides MINLP models, translates all other GAMS model types (e.g. LP, MCP, MIP, NLP) into different formats. A variety of options allow a customized translation into a growing number of formats. Currently, these formats include AMPL [10], BARON [16], LINGO [14], LGO [15], and MINOPT [17]. Details about convert can found in the GAMS Solver Guide [6].

A translation server has been set up for people without access to a GAMS system. The interface to this server is email based. An email to gms2xx@gamsworld.org with a GAMS model attached and the requested language name (e.g. AMPL) in the subject line triggers a translation process on this server. The translated model together with a translation report is mailed back to the sender. In case of a problem (e.g. a compilation error due to incorrect GAMS syntax) an error report is generated and mailed back to the sender. The usage of the translation server may require some preparatory work on the model to be sent. The server expects a single GAMS source file, hence complex models with a nested file structure must be merged. The provided services are limited to translation, which means that real solver calls or executions of external programs cannot be part of the submitted GAMS program. Details about the prerequisites can be found at the translation server web site at http://www.gamsworld.org/translate.htm.

With minor restrictions, the translation server provides simple and free access to the GAMS translation solver convert. Complex translation jobs that do not fit within the limits of the translation server have to be performed by using convert in combination with a regular GAMS system.

2. MINLPLib

The models in MINLPLib vary from small scale literature models from [7] and [8] to large scale real world models from different application areas like Agricultural Economics, Chemical-, Civil-, and Electrical Engineering, Finance, Management and OR. MINLPLib is not a static collection of models but will hopefully benefit from contributions from

the academic as well as the commercial world. At the moment the library consists of 136 models. We have 13 artifical literature models and 123 models from 40 different applications. For 59 of 136 models the global minimum (all problems are minimization problems) is known. For 67 models we have some integer solution and for 10 models no integer solution is known. For 4 of those 10 we cannot even provide a solution to the relaxed problem. The size of the models varies from tiny models (e.g. *gear* with one equation, five variables of which four are integer) to huge models (e.g. *nuclear10b* with 24972 equations, 23827 variables of which 10920 are binary, and over 100.000 non zeros, 23252 of which are nonlinear). A complete statistic of the model sizes is part of MINLPLib.

We collected and partially translated the models from different sources. We harvested models from existing collections on the Web:

- GAMSLIB (http://www.gams.com/modlib/modlib.htm)
- MacMINLP (http://www.maths.dundee.ac.uk/~sleyffer/MINLP_TP)
- MINOPT library (http://titan.princeton.edu/MINOPT/library.html)
- Test Problems from [8] (http://titan.princeton.edu/TestProblems/chapter12.html)

However, the main contribution of the new library to the research community is the access to real world model developed in commercial organizations. Models resulting from academic research tend to solve easily (because quite some creative effort went into the development of the model) or are almost intractable for general-purpose MINLP solvers. In some cases commercial models represent a straight-forward implementation of a real world optimization problem which sometimes leaves room for creative but more often for pure mechanical reformulations and improvements. In general, MINLP solvers, unlike their linear relatives, have a lack of preprocessing techniques, which make quite a difference, in particular for the real world models. For some applications we have included a whole family of closely related models (e.g. the *nuclear* models) from straight-forward formulations to manually preprocessed versions of the model. We also have completely different formulations for the same problem (e.g. *trim loss* models).

The models of MINLPLib are intended to serve as test models for MINLP algorithms, so the library intentionally includes badly formulated and even infeasible models. This variety should help test the reliability of MINLP solvers in an extreme but very practical environment.

MINLPLib is distributed as a ZIP file via the Web site. The distribution includes the scalar GAMS models, a collection of points for each model, and GAMS programs that manage meta information about the models (e.g. references to publications, contributors, etc.). The models are identified by their unique name (e.g. batch.gms), are in the scalar format described in section 1, and include a header with some model size statistics.

The point data consist of primal and dual values representing solutions to the model or its relaxed version. The header of a file containing a point might include a description of the point with references to the contributor, the algorithm or sequence of algorithms applied to find the point, the type of solution (integer or relaxed), and other useful information. The points are identified by the model name plus an extension .p1, p2, … (e.g. batch.p1). The scalar GAMS models in MINLPLib are set up in a way that a particular point can easily be loaded before the solve (or translation) statement by passing its name through the user1 GAMS parameter. For example, a GAMS command line to solve the model batch.gms starting from point batch.p1 would be: gams batch user1=batch.p1. An MINLP solver can make use of a point in different ways. For NLP-based MINLP solvers a point might result in an advanced basis that warm-starts the NLP solver. A solver might try to check the starting point for feasibility, hence a point that represents a solution to the MINLP would quickly result in an integer solution which might lead to some bounds used for faster termination (e.g. for branch-and-bound based methods).

MINLPLib also contains some GAMS programs (minlp*.gms) that help organize the MINLP models and points. These programs provide proper mappings of models to points, contributors, publications, and applications. The program minlpscript.gms contains some examples for creating batch execution scripts, so it is easy to create a customized script that, for example, runs all “demo size” trim loss minimization models in MINLPLib starting from their best known solution.

3. Conclusions

MINLPLib is part of a larger MINLP initiative that has its home at the MINLP World web site at http://www.gamsworld.org/minlp. The initiative tries to create a forum for discussion and dissemination of information about all aspects of MINLP. The purpose of the MINLP World site is to bring people who work with MINLP together. The site includes information about software for MINLP, the MINLPLib, and links to relevant sites, and it tries to improve open discussion by hosting a list server dedicated to MINLP. It is our hope that the development around MINLP will prosper and that the new library together with the translation service will help overcome the lack of interesting test problems available in different environments.

We would like to thank the developers of the existing MINLP collections: GAMSLIB, MacMINLP, MINOPT Library, Test Problems in [7] and [8] and our clients, who gave us permission to publish their models in a scalar form. These models form the backbone of MINLPLib.

```
Set I Products         /P1*P2/
    J Cutting Patterns /C1*C2/;

Parameter c(J)    cost of raw material                 /C1 1, C2 1/
          cc(J)   cost of change-over of knives        /C1 0.1, C2 0.2/
          b(I)    width of product roll-type I         /P1 460, P2 570/
          nord(I) number of orders of product type I   /P1 8, P2 7/
          Bmax    width of raw paper roll              /1900/
          Delta   tolerance for width                  / 200/
          Nmax    max number of products in cut        /   5/
          bigM    max number of repeats of any pattern /  15/;

Variable  y(J)    cutting pattern
          m(J)    number of repeats of pattern j
          n(I,J)  number of products I produced in cut J
          obj     objective variable;

Binary Variable y; Integer Variable m, n;

Equation defobj, max_width(J), min_width(J), max_n_sum(J),
                 min_order(I), cut_exist(J), no_cut(J);

defobj..         sum(j, c[j]*m[j] + cc[j]*y[j]) =e= obj;
max_width(j)..   sum(i, b[i]*n[i,j])            =l= Bmax;
min_width(j)..   sum(i, b[i]*n[i,j]) + Delta    =g= Bmax;
max_n_sum(j)..   sum(i, n[i,j])                 =l= Nmax;
min_order(i)..   sum(j, m[j]*n[i,j])            =g= nord[i];
cut_exist(j)..   y[j]                           =l= m[j];
no_cut(j)..      m[j]                           =l= bigM*y[j];

m.up[j] =  bigM; n.up[i,j] = nmax;

model trimloss /all/;
solve trimloss minimize obj using minlp;
```

Figure 1

```
*  MINLP written by GAMS Convert

Variables  b1,b2,i3,i4,i5,i6,i7,i8,x9;

Binary Variables b1,b2;
Integer Variables i3,i4,i5,i6,i7,i8;

Equations  e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,
           e11,e12,e13;

e1..  0.1*b1 + 0.2*b2 + i3 + i4 - x9 =E= 0;
e2..  460*i5 + 570*i7 =L= 1900;
e3..  460*i6 + 570*i8 =L= 1900;
e4..  460*i5 + 570*i7 =G= 1700;
e5..  460*i6 + 570*i8 =G= 1700;
e6..  i5 + i7 =L= 5;
e7..  i6 + i8 =L= 5;
e8..  i3*i5 + i4*i6 =G= 8;
e9..  i3*i7 + i4*i8 =G= 7;
e10.. b1 - i3 =L= 0;
e11.. b2 - i4 =L= 0;
e12.. - 15*b1 + i3 =L= 0;
e13.. - 15*b2 + i4 =L= 0;

* set non default bounds
i3.up = 15; i4.up = 15; i5.up = 5;
i6.up = 5;  i7.up = 5;  i8.up = 5;

Model m / all /;
Solve m using MINLP minimizing x9;
```

Figure 2

```
#  MINLP written by GAMS Convert

var b1 binary;
var b2 binary;
var i3 integer >= 0, <= 15;
var i4 integer >= 0, <= 15;
var i5 integer >= 0, <= 5;
var i6 integer >= 0, <= 5;
var i7 integer >= 0, <= 5;
var i8 integer >= 0, <= 5;
var x9;

minimize obj: x9;

subject to

e1:  0.1*b1 + 0.2*b2 + i3 + i4 - x9 = 0;
e2:  460*i5 + 570*i7 <= 1900;
e3:  460*i6 + 570*i8 <= 1900;
e4:  460*i5 + 570*i7 >= 1700;
e5:  460*i6 + 570*i8 >= 1700;
e6:  i5 + i7 <= 5;
e7:  i6 + i8 <= 5;
e8:  i3*i5 + i4*i6 >= 8;
e9:  i3*i7 + i4*i8 >= 7;
e10: b1 - i3 <= 0;
e11: b2 - i4 <= 0;
e12: - 15*b1 + i3 <= 0;
e13: - 15*b2 + i4 <= 0;
```

Figure 3

References:

1. R.E. Bixby, S. Ceria, C.M. McZeal, M.W.P. Savelsbergh, 1998. An Updated Mixed Integer Programming Library. MIPLIB 3.0, Optima 58, p.12-15.

2. Brooke, A. Drud, and A. Meeraus, 1984, High Level Modeling Systems and Nonlinear Programming, p 178-198 in P.T. Boggs et.al. Numerical Optimization 1984, SIAM.

3. Brooke, D. Kendrick, and A. Meeraus, 1988. GAMS: A Users Guide, The Scientific Press.

4. Conn, A.R., N.I.M Gould, and P.L. Toint: The SIF Reference Document at: http://www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html.

5. CUTE: Constrained and Unconstrained Testing Environment at: http://www.cse.clrc.ac.uk/Activity/CUTE.

6. GAMS Development Corp. Washington, D.C., 2001: GAMS - The Solver Manuals.

7. C.A. Floudas and P.M. Pardalos, 1990. A Collection of Test Problems for Constrained Global Optimization Algorithms, Lecture Notes in Computer Science no 455, Springer-Verlag, 1990

8. C.A. Floudas et.al., 1999. Handbook of Test Problems in Local and Global Optimization, Nonconvex Optimization and its Applications, vol. 33, Kluwer Academic Publishers.

9. C.A. Floudas: MINOPT Model Library at: http://titan.princeton.edu/MINOPT/library.html

10. R. Fourer, D.M. Gay, and B.W. Kernighan, 1993. AMPL: A Modeling Language for Mathematical Programming, The Scientific Press.

11. Fragniere E., J. Gondzio R. Sarkissian and J.-P. Vial, Structure Exploiting Tool in Algebraic Modeling Languages, Logilab Technical Report 97.2, Department of Management Studies, University of Geneva, Switzerland, June 1998.

12. D.M. Gay, 1985. Electronic Mail Distribution of Linear Programming Test Problems, Mathematical Programming Society COAL Newsletter.

13. Leyffer, S: Collection of MINLP models at: http://www.maths.dundee.ac.uk/~sleyffer/MINLP_TP/index.html

14. LINDO Systems Inc., Chicago, 1999: LINGO Users Guide.

15. Pinter, J.D., 1999, LGO: An integrated model development and solver system for continuous global optimization, *INFORMS Computing Society Newsletter* 20 (1), 1999.

16. Ryoo, H. S., and N. V. Sahinidis, 1996. A Branch-and-Reduce Approach to Global Optimization, *Journal of Global Optimization*, 8(2), 107-139.

17. Schweiger, C.A. and C.A. Floudas, MINOPT - A Modeling Language and Algorithmic Framework for Linear, Mixed-Integer, Nonlinear, Dynamic, and Mixed-Integer Nonlinear Optimization, Department of Chemical Engineering, Princeton University, 1998.