

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

Michael R. Bussieck • Arne Stolbjerg Drud • Alexander Meeraus

*GAMS Development Corp, 1217 Potomac Street, NW, Washington, D.C. 20007, USA*

*ARKI Consulting & Development A/S, Bagsvaerdvej 246A, 2880 Bagsvaerd, Denmark*

*GAMS Development Corp, 1217 Potomac Street, NW, Washington, D.C. 20007, USA*

[MBussieck@gams.com](mailto:MBussieck@gams.com) • [ADrud@arki.dk](mailto:ADrud@arki.dk) • [AMeeraus@gams.com](mailto:AMeeraus@gams.com)

---

The paper describes a new computerized collection of test models for mixed-integer nonlinear programming. Because 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 other formats, including AMPL, BARON, LGO, LINGO, and MINOPT. The translation server can also be used to transform industrial models that contain confidential information. Such transformations allow many of these models to be distributed to the research community as highly relevant algorithmic test models.

*(Mixed-Integer Nonlinear Programming, Testing, Modeling Systems)*

---

## 1. 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 and software developers, access to a wide set of interesting models with different characteristics has always been important. The lack of such models can hold back creative work. In 1985, when David Gay started the Netlib collection of linear programming (LP) models in the industry-standard MPS format (Gay 1985), he created an institution that has had an important impact on the LP community. Today, all papers describing new developments in LP refer in some form to experiments with some of these models, and an LP code is not considered acceptable unless it performs reasonably well on the Netlib collection. The more

recently developed MIPLIB collection (Bixby et al. 1998) 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., Floudas and Pardalos 1990, Floudas et al. 1999), and some of these models are MINLP models. Web sites like <http://titan.princeton.edu/MINOPT/library.html> (Floudas) and [http://www.maths.dundee.ac.uk/~sleyffer/MINLP\\_TP/index.html](http://www.maths.dundee.ac.uk/~sleyffer/MINLP_TP/index.html) (Leyffer) offer models used to test their own codes, some written in GAMS and others in MINOPT and AMPL. But the main reason we do not yet have a common computerized collection of models is that we do not yet use a common format, like the MPS format, to represent our underlying nonlinear models.

Over the years, many attempts have been made to create a standard format for nonlinear programming (NLP). One example is the SIF format defined at <http://www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html>, and used in the CUTE collection at <http://www.cse.clrc.ac.uk/Activity/CUTE>. However, none of these proposed formats has been widely accepted. In particular, practical modelers have not accepted them, primarily because the type of information required by different NLP algorithms is simply too different (Brooke et al. 1984). Today, practical NLP and MINLP models are therefore often represented in modeling system languages, such as GAMS (Brooke et al. 1988) or AMPL (Fourer et al. 1993), which 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 from which to choose, 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 are often based on confidential data or proprietary information. Such privately-developed models often have a different structure than university models, and therefore can be very useful to algorithm developers. Unfortunately, we seldom have access to these 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" (<http://www.gamsworld.org/minlp>). We have attacked problems both of format and of confidentiality by using a new model translator built into the GAMS system. This translator and its output are described in Section 2. Information about the MINLP library ("MINLPLib") is provided in Section 3, and Section 4 contains additional information about the MINLP initiative, and presents our conclusions.

## 2. CONVERT - A GAMS Model Translator

Today, most practical models involving nonlinearities are 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 as the basis of our collection. Since some MINLP solvers do not have an interface to GAMS, the model library described in this article will only be of general use unless we can also 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, our approach has been 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., Fragnière et al. 2000), such structures are not used by the solvers. Most solvers see the world as a list of variables,  $X_1$  to  $X_n$ , a list of equations or constraints,  $E_1$  to  $E_m$ , and the relationship between these variables and equations, as represented in some form. As long as the model seen by the solver remains unchanged, it is therefore acceptable for a translator to remove the structure.

The GAMS translator CONVERT 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 the following:

- Declarations of the variables, with extra declarations for the subsets of positive, integer, or binary variables,
- Declarations of the equations,
- 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 another language's format. For example, 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. There are a few cases where 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.

GAMS models have an objective variable rather than an objective function. If there is one defining equation for a given variable, CONVERT will eliminate the objective variable and will introduce a *real* objective function for formats that support objective functions (e.g. AMPL). Figure 1 is a user-written GAMS model for trim-loss minimization, and Figures 2 and 3 show this model's scalar GAMS and AMPL versions.

```

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: User-Written GAMS Model for Trim-Loss Minimization

```

* 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: Scalar GAMS Version of Figure 1

```

# 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;

minimize obj: 0.1*b1 + 0.2*b2 + i3 + i4;
subject to

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: Scalar AMPL Version of Figure 1

MINLP World seeks to provide examples of good model formulation that can best be viewed in the original algebraic presentation. Links to the original AMPL, GAMS, or MINOPT formulation are therefore part of MINLP World's web-provided, non-proprietary models.

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 remains is the size of the model, the structure of the Jacobian, and some linear and nonlinear relationships with numerical coefficients. Because the developer of a model is often unable to recognize his or her own model, industrial users with an interest in improving the solvers they use can, in many cases, be allowed to release the scalar version of their model to a model library like MINLPLib. Indeed, many of the initial models in the library are of industrial origin, and we expect that the number of these models will grow considerably in the future.

The GAMS translator is implemented as CONVERT, a regular GAMS solver. CONVERT is part of any GAMS system. In addition to MINLP models, CONVERT translates all other GAMS model types (e.g., LP, MCP, MIP, NLP) into different formats. A variety of options allows customized translation into a growing number of formats, which currently include AMPL (Fourer et al. 1993), BARON (Ryoo and Sahinides 1996), LINGO (LINDO Systems 1999), LGO (Pintér

1999), and MINOPT (Schweiger and Floudas 1998). Details about CONVERT can be found in the GAMS Solver Guide (GAMS Development Corp. 2001).

We have set up a translation server for people without access to a GAMS system. The interface to this server is e-mail-based. Sending an e-mail to [gms2xx@gamsworld.org](mailto:gms2xx@gamsworld.org) with an attached GAMS model and the requested language name in the subject line (e.g., AMPL) triggers a translation process on this server. The translated model, together with a translation report, is mailed back to the sender. If any problem occurs (e.g., a compilation error due to incorrect GAMS syntax), an error report is generated and mailed back to the sender. Using this 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 these and other prerequisites can be found at the translation server web site, <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 must be performed by using CONVERT in combination with a regular GAMS system.

### **3. MINLPLib**

The models in MINLPLib vary from small scale literature models (Floudas and Pardalos 1990, Floudas et al. 1999) to large-scale real-world models from different application areas, such as agricultural economics; chemical, civil, and electrical engineering, finance, management, and operations research. MINLPLib is not intended to be a static collection of models. Our hope is that it will benefit from contributions from both the academic and the commercial worlds.

The library currently contains 136 models, including 13 artificial literature models and 123 models from 40 different applications. In 59 of the 136 models, the global minimum is known (original maximization problems have been converted to minimization problems). 67 models have some integer solution, and ten models have no known integer solution (for four of these ten models, we cannot even provide a solution to the relaxed problem). The size of the models varies from tiny (e.g., gear with one equation and five variables, of which four are integer) to huge (e.g., nuclear10b with 24972 equations and 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 library's models from different sources, including models from the following existing collections on the Web:

- GAMSLIB (<http://www.gams.com/modlib/modlib.htm>)
- MacMINLP ([http://www.maths.dundee.ac.uk/~sleyffer/MINLP\\_TP](http://www.maths.dundee.ac.uk/~sleyffer/MINLP_TP))
- MINOPT library (<http://titan.princeton.edu/MINOPT/library.html>)
- Test Problems from Floudas et al. (1999)  
(<http://titan.princeton.edu/TestProblems/chapter12.html>)

We believe the main contribution of this new library to the research community is access to real-world models developed by commercial organizations. Models resulting from academic research tend to solve easily (because a great deal of creative effort went into their development), 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. These models can leave room for creativity, but most often support pure, mechanical reformulations and improvements. In general, and unlike their linear relatives, MINLP solvers lack preprocessing techniques. This can make quite a difference, in particular for the real-world models.

For some applications, we have included an entire family of closely related models (e.g., the nuclear models), from straight-forward formulations to manually preprocessed versions. We have also provided completely different formulations for the same problem (e.g., trim-loss models).

Because we intend the models contained within MINLPLib to serve as test models for MINLP algorithms, we have intentionally included some badly formulated, and even infeasible, models. Such variety should help test the reliability of MINLP solvers in an extreme, but very practical, environment.

MINLPLib is distributed as a ZIP file at [www.gamsworld.org/minlp/minlplib.htm](http://www.gamsworld.org/minlp/minlplib.htm). 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.). Each model is identified by a unique name (e.g. "batch.gms"), is provided in the scalar format described in Section 1, and includes a header that contains some model-size statistics.

Included 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 may include a description of that point, and may provide 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 a ".p#" extension (e.g., batch.p1).

We have organized the scalar GAMS models in MINLPLib so 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, the 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 that could 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 MINLPLib and its corresponding web site, MINLP World. Like some other modeling languages, the GAMS language provides more than just statements for mathematical-model generation. GAMS programs can be used to store, manage, and process structured information, such as the information organized in a database system. Hence, information about models, points, contributors, publications, and applications are stored and maintained in the GAMS program `minlplib.gms`.

Because the program `minlpscript.gms` contains examples for creating batch execution scripts, it is easy to create a customized script, for example, that runs all “demo-size” trim-loss-minimization models in MINLPLib, starting from their best known solution.

## 4. 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>. This initiative tries to create a forum for discussion and the dissemination of information about all aspects of MINLP.

Our goal in creating 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. It attempts to improve open discussion by hosting a list server dedicated to MINLP. It is our hope that development efforts around MINLP will prosper through the use of this new library and its associated translation service. MINLPLib will, we trust, help all MINLP developers overcome the current lack of interesting test problems available in different environments.

## Acknowledgments

We would like to thank the developers of the existing MINLP collections GAMS LIB, MacMINLP, and collections by Floudas et al., including the MINOPT Library. We also thank our clients, who gave us permission to publish their models in a scalar form. These models form the backbone of MINLPLib.

## References

Bixby, R.E., S. Ceria, C.M. McZeal, M.W.P. Savelsbergh. 1998. An updated mixed integer programming library MIPLIB 3.0. *Optima* **58** 12-15.

Brooke, A.; A. Drud, A. Meeraus. 1984. High level modeling systems and nonlinear programming. P.T. Boggs, R.H. Byrd, R.B. Schnabel, editors . *Numerical Optimization 1984*. SIAM, Philadelphia, PA. 178-198.

Brooke, A., D.Kendrick, A. Meeraus. 1988. *GAMS: A Users Guide*. The Scientific Press, San Francisco, CA.

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

Floudas, C.A., P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer, C.A. Schweiger., 1999. *Handbook of Test Problems in Local and Global Optimization, Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

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

Fraginière, E., J. Gondzio, R. Sarkissian, J.-P. Vial. 2000. A Structure Exploiting Tool in Algebraic Modeling Languages. *Management Science* **46** 1145-1158.

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

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

LINDO Systems Inc. 1999: *LINGO Users Guide*. LINDO Systems Inc., Chicago, IL.

Pintér, J.D. 1999. LGO: an integrated model development and solver system for continuous global optimization. *INFORMS Computing Society Newsletter* **20** 1999.

Ryoo, H.S., N. V. Sahinidis. 1996. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*. **8** 107-139.

Schweiger, C.A., and C.A. Floudas. 1998. MINOPT - a modeling language and algorithmic framework for linear, mixed-integer, nonlinear, dynamic, and mixed-integer nonlinear optimization. Department of Chemical Engineering, Princeton University, Princeton, NJ.