LaGO - An object oriented library for solving MINLPs[†]

Ivo Nowak[‡], Hernán Alperin, and Stefan Vigerske

Humboldt-Universität zu Berlin
Institut für Mathematik
Rudower Chaussee 25,
D-12489 Berlin, Germany
http://www-iam.mathematik.hu-berlin.de/~eopt/

Abstract. The paper describes a software package called LaGO for solving nonconvex mixed integer nonlinear programs (MINLPs). The main component of LaGO is a convex relaxation which is used for generating solution candidates and computing lower bounds of the optimal value. The relaxation is generated by reformulating the given MINLP as a block-separable problem, and replacing nonconvex functions by convex underestimators. Results on medium size MINLPs are presented.

Keywords. mixed integer nonlinear programming, convex relaxation, heuristics, decomposition, software

AMS classifications. 90C22, 90C20, 90C27, 90C26, 90C59

1 Introduction

Several strategies for solving nonconvex MINLPs have been proposed [BP03]. Exact solution approaches include branch-and-reduce [TS02], branch-and-bound [ADFN98,SP99], interval analysis [VEH96] and outer approximation [KAGB01]. On the other side, heuristic solution algorithms explicitely addressing MINLP, such as the multistart scatter search heuristic [ULP$^+$02], appeared rather rarely in the literature. More information on global optimization algorithms can be found in [Flo00,HP95,BSV$^+$01].

LaGO (**La**grangian**G**lobal**O**ptimizer) is an object oriented library for solving nonconvex mixed integer nonlinear programs (MINLPs) written in `C++`. The basic component of the solver is a relaxation, which is used for generating solution candidates and computing lower bounds of the optimal value. It is very important that the quality of the relaxation is good enough, in the sense that the amount of work to retrieve a good solution from the relaxation is not too high. For improving a given relaxation, a series of operations, which can be performed

[†] The work was supported by the German Research Foundation (DFG) under grant NO 421/2-1.

[‡] `ivo@mathematik.hu-berlin.de`, corresponding author

relatively fast and take advantage of a partially separable structure of the given optimization problem, is used.

The software can be used as a general purpose MINLP solver or as a toolbox for developing specialized solution algorithms. The object oriented design of the library allows easy replacement of modules, such as linear algebra routines and various solvers. Optimization problems can be either provided in AMPL-format [FGK93] or coded in `C++` using LaGO 's class `MinlpProblem`. The solver supports two types of structural properties: *partial separability*, i.e. the Hessians have almost block-diagonal structure, and *sparse and low rank Hessians*, i.e. there exist fast methods for multiplying a vector with a Hessian.

The basic components of the current version of LaGO are: *preprocessing*, *convex relaxation* and *relaxation-based heuristic*, which are described in Sections 2, 3 and 4. A *branch-and-cut algorithm* is currently under development and will be added in the future. In Section 5 we report preliminary results, and give conclusions in Section 6.

Notation. Let $J \subset \{1, \dots, n\}$ be an index set. We denote by $|J|$ the number of elements of J. A subvector $x_J \in \mathbb{R}^{|J|}$ of $x \in \mathbb{R}^n$ is defined by $(x_j)_{j \in J}$. Similarly, a function $f_J(x)$ is defined by $(f_j(x))_{j\in J}$. The space of k times differentiable functions $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is denoted by $C^k(\mathbb{R}^n, \mathbb{R}^m)$.

2 Preprocessing

We assume that the given MINLP problem has the form:

$$(\mathrm{P}) \qquad \begin{array}{ll} \min & h_0(x) \\ \text{s.t.} & h_I(x) \le 0 \\ & h_E(x) = 0 \\ & x \in Y \end{array}$$

where
$$Y = \{x \in [\underline{x}, \overline{x}] \mid x_j \in \{\underline{x}_j, \overline{x}_j\} \text{ for } j \in B\},$$
$\underline{x}, \overline{x} \in \mathbb{R}^n$, $B \subseteq \{1, \dots, n\}$, $I \cup E = \{1, \dots, m\}$ with $I \cap E = \emptyset$, and $h_i \in C^2(\mathbb{R}^n, \mathbb{R})$ for $i = 0, \dots, m$. Note that MINLPs with piecewise twice differentiable functions and integrality constraints, $x_i \in [\underline{x}_i, \overline{x}_i] \cap \mathbb{Z}$, can be transformed into the form (P) by introducing additional binary variables.

To facilitate the construction of a relaxation, problem (P) is transformed into the following *block-separable* extended reformulation.

$$(\mathrm{P}_{\mathrm{ext}}) \qquad \begin{array}{lll} \min & c^T x + c_0 & \\ \text{s.t.} & A_I x + b_I \le 0 & \\ & A_E x + b_E = 0 & \\ & g^k(x_{J_k}) \le 0, & k = 1, \dots, p \\ & x \in Y & \end{array}$$

where $\{J_1, \dots, J_p\}$ is a partition of $\{1, \dots, n\}$, i.e. $\bigcup_{k=1}^p J_k = \{1, \dots, n\}$ and

$J_i \cap J_k = \emptyset$ for $i \neq k$, $g^k \in C^2(\mathbb{R}^{n_k}, \mathbb{R}^{m_k})$, $k = 1, \dots, p$, with $n_k = |J_k|$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A_I \in \mathbb{R}^{(|I|,n)}$ and $A_E \in \mathbb{R}^{(|E|,n)}$. To obtain the block functions $g^k(x_{J_k})$ used in (P$_{\text{ext}}$) we construct the *sparsity graph* $G_s = (V, E_s)$ with the nodes $V = \{1, \dots, n\}$ and the edges

$$E_s = \Big\{(i,j) \in V^2 \;\Big|\; \frac{\partial^2 h_k(x)}{\partial x_i \partial x_j} \neq 0 \text{ for some } k \in \{0, \dots, m\} \text{ and } x \in [\underline{x}, \overline{x}]\Big\}.$$

Let $\{\tilde{J}_1, \dots, \tilde{J}_p\}$ be a partition of V. We define the set of nodes of $\bigcup_{l=k+1}^p \tilde{J}_l$ connected to $\tilde{J}_k$ by $R_k = \{i \in \bigcup_{l=k+1}^p \tilde{J}_l \mid (i,j) \in E_s,\, j \in J_k\}$, for $k = 1, \dots, p$. The set R_k can be interpreted as the set of flows of a nonlinear network problem connecting a component $\tilde{J}_k$ with components $\tilde{J}_l$, where $k < l$. After introducing new variables $y^k \in \mathbb{R}^{|R_k|}$ for every set R_k, and using the copy-constraints $x_{R_k} = y^k$, problem (P) is formulated as a block-separable program with respect to the blocks $J_k = (\tilde{J}_k, R_k)$, with $k = 1, \dots, p$. Finally, nonlinear equality constraints $h_i(x) = 0$ for $i \in E$ are replaced by the two inequality constraints $h_i(x) \leq 0$ and $-h_i(x) \leq 0$, and block-separable constraints

$$\sum_{k=1}^p h_i^k(x_{J_k}) \leq 0$$

are replaced by

$$\sum_{k=1}^p t_{ik} \leq 0, \quad h_i^k(x_{J_k}) \leq t_{ik}, \quad k = 1, \dots, p.$$

The resulting program has the form (P$_{\text{ext}}$), with $g_i^k(x_{J_k}, t_{ik}) = h_i^k(x_{J_k}) - t_{ik}$. For the sake of simplicity we keep the notation n, m, B, I, E, Y as in (P) including the new variables t_{ik} into the block x_{J_k}.

Furthermore, the type (linear, convex, concave) of all functions is determined by evaluating the minimum and maximum eigenvalue of each Hessian at sample points. Since the given MINLP is coded in AMPL [FGK93], all functions are given in a black-box representation, i.e. there exist only procedures for evaluating functions, gradients and Hessians. Therefore, the generation of the sparsity graph and the determination of function types is performed by sampling techniques.

Problem (P) assumes the existence of bounds $\underline{x}_j$ and $\overline{x}_j$ for all $j = 1, \dots, n$. This is not the usual case in the instances collected from MINLPLib [BDM03] for instance. Since we need the box $[\underline{x}, \overline{x}]$ to compute convex relaxations and for the above sampling technique, we generate missing bounds by determining the maximum and minimum value of a variable over a domain defined by the convex constraints of problem (P). If there are not enough convex constraints to determine a variable bound, we use a simple guessing algorithm.

3 Convex relaxation

A *convex relaxation* of problem (P$_{\text{ext}}$) is defined by

$$
\text{(C)} \qquad \begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & A_I x + b_I \le 0 \\ & A_E x + b_E = 0 \\ & \breve{g}^k(x_{J_k}) \le 0, \qquad k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}] \end{array}
$$

where $\breve{g}^k$ are *convex underestimators* of g^k over $X_k = [\underline{x}_{J_k}, \overline{x}_{J_k}]$, i.e. $\breve{g}^k(x) \le g^k(x)$ for all $x \in X_k$ and $\breve{g}^k$ is convex over X_k. The best convex underestimators are convex envelopes. Since computing convex envelopes of nonconvex functions can be computationally very expensive, we use the following procedure for generating convex relaxations.

We construct at first a *polynomial relaxation* of $(\mathrm{P}_{\mathrm{ext}})$ by replacing non-quadratic functions by *polynomial underestimators.* We assume that the non-quadratic functions of $(\mathrm{P}_{\mathrm{ext}})$ are of the form $g(x) = b^T x + f(x_N)$ where the number $|N|$ of nonlinear variables is small. We use as an underestimator for f a multivariate polynomial defined by

$$
p(x) = a^T \varphi(x) \tag{1}
$$

where $\varphi(x) = (x^{\beta_1}, \ldots, x^{\beta_r})^T$, $\beta_j \in \mathbb{N}_0^n$, for $1 \le j \le r$, is a vector of monomials, and $a \in \mathbb{R}^r$ is the vector of coefficients for each monomial. The degree of the polynomial p is the number $d = \max_{j=1}^r |\beta_j|$ with $|\beta| = \sum_{i=1}^n \beta_i$. For our published results, we use $d = 2$, but in our implementation larger degree polynomial can be used. Let D^k be a differential operator defined by

$$
D^k(f(x)) = \left(\frac{\partial^k f(x)}{\partial x^{\beta_l}} \right)_{1 \le l \le r_k},
$$

with $|\beta_l| = k$ for $l = 1, \ldots, r_k \le \binom{r}{k}$ can be at most all the possible monomials of degree k, but in fact the sparsity pattern is used and fewer monomial than $\binom{r}{k}$ are needed. In order to determine the coefficients $a \in \mathbb{R}^r$ of the polynomial underestimator (1), we solve the linear program

$$
\text{(U)} \qquad \begin{array}{ll} \displaystyle\min_{a \in \mathbb{R}^r} \sum_{k=0}^{2} \delta_k \sum_{x \in S_k} \| D^k(f(x) - a^T \varphi(x)) \|_1 & \\ \text{s.t.} \qquad\qquad\qquad a^T \varphi(x) \le f(x),\, x \in S_0. & \end{array}
$$

The coefficients $\delta_0 > \delta_1, \delta_2$ give the relative importance to the information that comes from the evaluation of the function, the gradient and the Hessian respectively. The finite sample sets $S_k, k = 0, 1, 2$ contain sample points for the computation of the function, gradient and Hessian respectively. In general, this approach is only rigorous for certain kind of functions such as convex and concave.

The polynomial relaxation of $(\mathrm{P}_{\mathrm{ext}})$ can be reformulated by adding some extra variables and constraints as the following *mixed-integer quadratic program* (MIQQP)

$$
\text{(Q)}\qquad \begin{array}{ll}
\min & c^T x + c_0 \\
\text{s.t.} & A_I x + b_I \leq 0 \\
 & A_E x + b_E = 0 \\
 & q^k(x_{J_k}) \leq 0, \qquad k = 1, \ldots, p \\
 & x \in Y
\end{array}
$$

where q^k, $k = 1, \ldots, p$, are quadratic forms. We use two methods for convexifying (Q).

The first method is based on replacing all nonconvex quadratic forms by the so-called α-underestimators introduced by Adjiman and Floudas [AF97]. An α-underestimator of a function $f \in C^2(\mathbb{R}^n, \mathbb{R})$ is the function

$$\breve{f}(x) = f(x) + \alpha^T r(x)$$

where

$$r(x) = \operatorname{Diag}(x - \underline{x})(x - \overline{x}) \tag{2}$$

and $\operatorname{Diag}(\cdot)$ denotes a diagonal matrix. The parameter $\alpha \in \mathbb{R}^n$ is computed according to

$$\alpha = \frac{1}{2} \max\{0, -\rho\} \operatorname{Diag}(w)^{-2} e$$

where $e \in \mathbb{R}^n$ is the vector of ones, $w = \overline{x} - \underline{x}$ is the diameter vector of the interval, and

$$\rho \leq \lambda_1(\operatorname{Diag}(w) \nabla^2 f(x) \operatorname{Diag}(w)), x \in [\underline{x}, \overline{x}],$$

is a bound on the the minimum eigenvalue of the transformed Hessian of f over $[\underline{x}, \overline{x}]$. If f is a quadratic form, i.e. $\nabla^2 f$ is constant, ρ can be determined by eigenvalue computation. In the general case it can be computed by interval Hessians [AF97] or using a sampling technique. It is clear that $\breve{f}(x) \leq f(x)$ for all $x \in [\underline{x}, \overline{x}]$, and $\breve{f}$ is convex. Note that $f = \breve{f}$ if f is convex.

Applying the α-underestimator technique directly to the original functions would also give a convex relaxation. However, our method is often tighter because the α-convexification depends only on the curvature of the function and not on the function behaviour. For more clarification see the example in Figure 1 where f is the original function, $\breve{f}$ the α-convexification of f, q the polynomial underestimator, and $\breve{q}$ the α-convexification of q.

The second method for convexifying problem (Q) is based on replacing the box-constraints $x_C \in [\underline{x}_C, \overline{x}_C]$ and the binary constraints $x_B \in \{\underline{x}_B, \overline{x}_B\}$ by the quadratic constraints $r_C(x) \leq 0$, and $r_B(x) = 0$ respectively, where the quadratic form r is defined as above, and $C = \{1, \ldots, n\} \setminus B$ is the index set of continuous variables. The resulting reformulation of (Q) is all-quadratic, and its dual is a *semidefinite program.* There exist several methods for solving semidefinite programs [WSV00]. We use a method based on formulating the dual as an eigenvalue optimization problem [Now02]. This approach allows fast computation of near optimal dual solutions and supports decomposition. For each dual point μ produced by this algorithm, the Lagrangian is convex, and defines the following convex *Lagrangian relaxation*

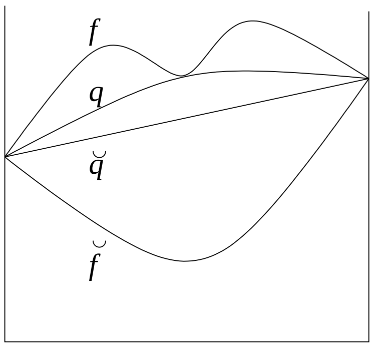

Fig. 1. α-convex underestimator versus the convexification of the polynomial underestimator.

$$(\mathrm{R}_\mu) \qquad \min_{x \in [\underline{x}, \overline{x}]} L(x; \mu),$$

where $L(\cdot; \mu)$ is the Lagrangian to the all-quadratic programming reformulation of (Q).

Besides of the above convexification methods, one further method for constructing a convex relaxation (C) is implemented. It uses quadratic *convex global underestimators* proposed by Phillips, Rosen and Walke [PRW95]. Their method for approximating the convex envelope of a nonlinear function f by a quadratic function was improved in LaGO to reduce the absolute value of the smallest eigenvalue of the obtained quadratic underestimator. Let $S \subset [\underline{x}, \overline{x}]$ be a finite sample set, and define the quadratic function

$$q(x; a, b, c) = c + 2b^T x + x^T \operatorname{Diag}(a) x,$$

where $a, b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then $q(\cdot; a, b, c)$ is convex, if and only if $a \geq 0$. The tightest quadratic convex underestimator $q(\cdot; a, b, c)$ over the set S is provided by the program

$$(\mathrm{CGU}) \qquad \begin{array}{lll} \min\limits_{a,b,c} \sum\limits_{x \in S} f(x) - q(x; a, b, c) + \delta e^T a & & \\ \text{s.t.} \quad f(x) \geq q(x; a, b, c), & & \text{for all } x \in S \\ \qquad\quad a \geq 0. & & \end{array}$$

Since q depends linearly on a, b, c, problem (CGU) is a linear program. The term $\delta e^T a$ reduces the absolute value of the smallest eigenvalue of $\operatorname{Diag}(a)$ in the case where (CGU) is degenerated. The quality of these underestimators depends strongly on the sample set S. If S contains all minimizers of f over $[\underline{x}, \overline{x}]$, the bound is rigorous. Since we cannot guarantee to find all minimizers, this approach provides a heuristic underestimator.

4 Relaxation-based heuristic

For computing solution candidates of a MINLP a rounding heuristic was developed, which is based on a convex relaxation of the given problem. The heuristic works by computing a solution point of problem (C), and rounding some binary components of this point. The rounded variables are fixed and the restricted convex relaxation is solved. This procedure is repeated as long, as either the restricted convex relaxation is infeasible, or all binary variables are fixed. In the latter case a local search is started from the last solution of the restricted convex relaxation giving a solution candidate. The values of the binary variables are recursively switched and the whole process is repeated as long, as either all combinations of binary variables are searched, or the number of solution candidates exceeds a given number s_{max}.

```
RoundHeu:

$Z = \emptyset$;
FixVariables($0, \emptyset$);


FixVariables($y, J$):

if    $\breve{\Omega} \cap U_{y,J} = \emptyset$, then return;
if    $J = B$,
then Optimize($y$);
else begin
        get a solution $x$ of $(C_{y,B})$;
        get $K \subset B \setminus J$ where $\min\{x_j - \underline{x}_j, \overline{x}_j - x_j\}$ is small for $j \in K$;
        for $j \in K$, round $x_j$;
        repeat
                call FixVariables($x, J \cup K$);
                switch some components of $x_K$ to get a new binary combination of $x_K$;
        until   ($|Z| \geq s_{\max}$ or all combinations of $x_K$ are searched);
        end


Optimize($y$):

if    $|B| = n$,
then if $y \in \Omega$, then add $y$ to $Z$;
else get a solution candidate of $(P_{y,B})$ by using a local search starting from the
        solution of $(C_{y,B})$, and add the point to $Z$;
```

Fig. 2. Algorithm *rounding heuristic*

To describe this method more detailed, let us define the original problem and the relaxed problem with fixed binary variables

$$(P_{y.J}) \qquad\qquad \min\{c^T x \mid x \in \Omega \cap U_{y,J}\}$$

and

$$(C_{y,J}) \qquad\qquad \min\{c^T x \mid x \in \breve{\Omega} \cap U_{y,J}\}$$

where

$$U_{y,J} = \{x \in \mathbb{R}^n \mid x_J = y_J\},$$

$y \in [\underline{x}, \overline{x}]$, $J \subset \{1, \dots, n\}$, and Ω and $\breve{\Omega}$ are the feasible sets of problems $(\mathrm{P}_{\mathrm{ext}})$ and (C) respectively. The recursive algorithm described in Fig. 2 computes a set Z of binary feasible solution candidates of problem $(\mathrm{P}_{\mathrm{ext}})$.

In order to improve the simple method `Optimize` for solving the continuous nonlinear program $(P_{y,B})$, a neighbourhood search or a deformation heuristic, as described in [AN02], can be used.

5 Preliminary numerical results

We tested the described algorithm using a set of instances of MINLP that were obtained from the GAMS web-site [BDM03,CG02]. For the computation of the minimum eigenvalue and corresponding eigenvector we used the Lanczos method ARPACK++ [GS97]. The sequential quadratic programming code SNOPT [GMS97] is used for finding local solutions of nonlinear optimization problems. To analyse its performance, the code was run on a machine with a 1GHz Pentium III processor and 256 MB RAM. Table 2 reports the performance on a selected set of problems using Algorithm `RoundHeu`. The definition of each column is explained in Table 1. The limit on $|S_{\mathrm{f}}|$, the maximum number of feasible points explored was set to 50.

6 Conclusions

We described the `C++` library LaGO for solving general nonconvex MINLPs. Main features of LaGO are: (i) the objective and constraint functions can be given in a black-box formulation, i.e. it is only assumed that functions, gradients and Hessians can be evaluated; (ii) the given MINLP is automatically reformulated as a block-separable formulation giving the possibility to apply decomposition techniques; (iii) the object oriented design of the software allows easy extensions and modifications of the current code.

Preliminary results with a relaxation-based rounding heuristic were presented, demonstrating that the current version of LaGO is able to solve medium size MINLP instances in a reasonable time. The only two instances that were not solved to known global optimality were those were the limit of $|S_{\mathrm{f}}| \leq 50$ feasible solutions was reached.

Table 1. Descriptions of the columns of Table 2.

rel. error	$\lvert f_{\text{heu}} - f_{\text{best}}\rvert/(1 + \lvert f_{\text{best}}\rvert)$, where f_{heu}, and f_{best} are the objective value obtained with the heuristic and the best known objective value for the problem respectively.
heu. time	seconds required by Algorithm `RoundHeu` (without preprocessing) to obtain the heuristic solution x_{heu}, and its corresponding objective value f_{heu}.
k	$k \le 2^{\lvert B\rvert}$ number of fixed binary combinations explored
$\lvert S_{\text{f}}\rvert$	$\lvert S_{\text{f}}\rvert \le k$, number of feasible solutions obtained from each $(\mathrm{C}_{y,B})$
	description of the MINLP instances.
n	number of variables,
m	number of constraints,
$\lvert E\rvert$	number of equality constraints,
$\lvert C\rvert$	number of continuous variables,
$\lvert B\rvert$	number of binary variables,
$\lvert U\rvert$	number of unbounded variables ($U = \{i \mid \underline{x}_i = -\infty \text{ or } \overline{x}_i = +\infty\}$).

The performance of the described solution approach depends mainly on the quality of the relaxation. In order to improve the relaxation, and to search systematically for a global solution, a branch-and-cut algorithm with box-reduction is currently under development and will be added in the future.

References

[ADFN98] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. *Computers and Chemical Engineering*, pages (1137–1158), 1998.

[AF97] C. S. Adjiman and C. A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *J. of Global Opt.*, 9:23–40, 1997.

[AN02] H. Alperin and I. Nowak. Lagrangian Smoothing Heuristics for MaxCut. Technical report, HU–Berlin NR–2002–6, 2002.

[BDM03] M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib - A Collection of Test Models for Mixed-Iinteger Nonlinear Programming. *INFORMS J. Comput.*, 15(1), 2003.

[BP03] M.R. Bussieck and A. Pruessner. Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News.*, 2003.

[BSV$^+$01] C. Bliek, P. Spellucci, L. Vicente, A. Neumaier, L. Granvilliers, E. Monfroy, F. Benhamou, E. Huens, P. Van Hentenryck, D. Sam-Haroud, and B. Faltings. *COCONUT Deliverable D1, Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of The Art.* `http://www.mat.univie.ac.at/~neum/glopt.html`, 2001.

[CG02] GAMS Development Corp. and GAMS Software GmbH. MINLP World, 2002. http://www.gamsworld.org/minlp/.

Table 2. Algorithm performance on a selected set of described examples.

example	rel. error	heu. time	k	$\lvert S_\mathrm{f}\rvert$	n	m	$\lvert E\rvert$	$\lvert C\rvert$	$\lvert B\rvert$	$\lvert U\rvert$
alan	0	0.02	6	6	9	8	3	5	4	-
batch	0	57.13	84	50	47	74	13	23	24	-
batchdes	0	0.70	12	8	20	20	7	11	9	-
ex1221	0	0.04	4	4	6	6	3	3	3	-
ex1222	0	0.00	1	1	4	4	1	3	1	-
ex1223b	0	0.11	16	16	8	10	1	4	4	-
ex1224	0	13.32	81	45	12	8	3	4	8	-
ex1225	0	0.07	7	6	9	11	3	3	6	-
ex1226	0	0.03	6	5	6	6	2	3	3	-
ex1252	.03	365.73	121	50	40	44	23	25	15	-
gbd	0	0.01	3	3	5	5	1	2	3	-
sep1	0	0.11	3	3	30	32	23	28	2	-
synthes1	0	0.10	5	5	7	7	1	4	3	-
synthes3	0	2.04	20	20	18	24	3	10	8	-
elf	.40	52.22	118	50	55	39	7	31	24	30
ex3	0	1.53	4	4	33	32	18	25	8	15
ex4	0	31.69	22	20	37	31	1	12	25	7
fuel	0	0.56	4	4	16	16	7	13	3	6
gkocis	0	0.08	6	6	12	9	6	9	3	6
meanvarx	0	0.51	20	20	36	45	9	22	14	21
oaer	0	0.05	6	6	10	8	4	7	3	6
procsel	0	0.04	2	2	11	8	5	8	3	6
synheat	0	15.94	21	20	57	65	21	45	12	32
synthes2	0	0.26	12	12	12	15	2	7	5	2
24 examples,	22	solved								

[FGK93] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Duxbury Press, Brooks/Cole Publishing Company, 1993.

[Flo00] C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications.* Kluwer Academic Publishers, 2000.

[GMS97] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT 5.3 user's guide. Technical report, University of California, San Diego, Mathematics Department Report NA 97-4, 1997.

[GS97] F. Gomes and D. Sorensen. `ARPACK++`*: a* `C++` *Implementation of* `ARPACK` *eigenvalue package*, 1997. `http://www.crpc.rice.edu/software/ARPACK/`.

[HP95] R. Horst and P. Pardalos. *Handbook of Global Optimization.* Kluwer Academic Publishers, 1995.

[KAGB01] P. Kesavan, R. J. Allgor, E. P. Gatzke, and P. I. Barton. Outer Approximation Algorithms for Separable Nonconvex Mixed-Integer Nonlinear Programs. *submitted to Mathematical Programming*, 2001.

[Now02] I. Nowak. Lagrangian Decomposition of Mixed-Integer All-Quadratic Programs. Technical report, HU–Berlin NR–2002–7, 2002.

[PRW95] A. Phillips, J. Rosen, and V. Walke. Molecular structure determination by global optimization. In *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, volume 23, pages 181–198. 1995.

[SP99] E. M. B. Smith and C. C. Pantelides. A Symbolic Reformulation/Spatial Branch and Bound Algorithm for the Global Optimization of nonconvex MINLPs. *Computers and Chemical Engineering*, 23:457–478, 1999.

[TS02] M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002.

[ULP$^+$02] Zsolt Ugray, Leon Lasdon, John Plummer, Fred Glover, Jim Kelly, and Rafael Marti. A multistart scatter search heuristic for smooth NLP and MINLP problems. *http://www.utexas.edu/courses/lasdon/papers.htm*, 2002.

[VEH96] R. Vaidyanathan and M. EL-Halwagi. Global optimization of nonconvex MINLP's by interval analysis. In I. E. Grossmann, editor, *Global Optimization in Engineering Design*, pages 175–193. Kluwer Academic Publishers, 1996.

[WSV00] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming.* Kluwer Academic Publishers, 2000.