LGO

János D. Pintér, Pintér Consulting Services, Inc. Halifax, NS, Canada, B3M 1J2 jdpinter@hfx.eastlink.ca, http://www.dal.ca/~jdpinter

Contents

1 Introduction

1.1 The LGO Solver Suite

The *Lipschitz-Continuous Global Optimizer*[1] (LGO) serves for the analysis and global solution of general nonlinear programming (NLP) models. The LGO solver system has been developed and gradually extended for more than a decade and it now incorporates a suite of robust and efficient global and local nonlinear solvers. It can also handle small LP models.

GAMS/LGO integrates the following global scope algorithms:

- Branch-and-bound (adaptive partition and sampling) based global search (BB)
- Adaptive global random search (GARS)
- Adaptive multistart global random search (MS)

LGO also includes the following local solver strategies:

- Bound-constrained local search, based on the use of an exact penalty function (EPM)
- Constrained local search, based on a generalized reduced gradient approach (GRG).

The overall solution approach followed by GAMS/LGO is based on the seamless combination of the global and local search strategies. This allows for a broad range of operations. In particular, a solver suite approach supports the flexible usage of the component solvers: one can execute fully automatic (global and/or local search based) optimization, and can design customized interactive runs.

[1] Also see `http://www.dal.ca/~jdpinter/l_s_d.html`

GAMS/LGO does not rely on any sub-solvers, and it does not require any structural information about the model. It is particularly suited to solve even 'black box' (closed, confidential), or other complex models, in which the available analytical information may be limited. GAMS/LGO needs only computable function values (without a need for higher order analytical information). GAMS/LGO can even solve models having constraints involving continuous, but non-differentiable functions. Thus, within GAMS, LGO is well suited to solve DNLP models.

GAMS/LGO can also be used in conjunction with other GAMS solvers. For instance, the local solver CONOPT can be used after LGO is finished to verify the solution and/or to provide additional information such as marginal values. To call CONOPT, the user can specify the LGO solver option `callConopt`. See the LGO Options section for details.

The LGO solver suite has been successfully applied to complex, large-scale models both in educational/research and commercial contexts for over a decade. Tractable model sizes depend only on the available hardware, although LGO has a 2000 variable, 2000 constraint size limit.

1.2 Running GAMS/LGO

GAMS/LGO is capable of solving the following model types: LP, RMIP, NLP, and DNLP. If LGO is not specified as the default solver for these models, it can be invoked by issuing the following command before the solve statement:

```
option (modeltype) = lgo;
```

where `(modeltype)` stands for `LP, RMIP, NLP,` or `DNLP`.

2 LGO Options

GAMS/LGO works like other GAMS solvers, and many options can be set directly within the GAMS model. The most relevant GAMS options are `reslim, iterlim`, and `optfile`. A description of all available GAMS options can be found in the Chapter "Using Solver Specific Options". See the GAMS Solver Manuals.

If you specify "`<modelname>.optfile = 1;`" before the SOLVE statement in your GAMS model, GAMS/LGO will then look for and read an option file with the name lgo.opt (see "Using Solver Specific Options" for general use of solver option files). The syntax for the LGO option file is

```
optname = value
```

with one option on each line. For example, one can write

```
opmode = 1
```

This specifies LGO to use global branch and bound search and the built-in local search methods.

The GAMS/LGO options are divided into two main categories:

- General options
- Limits and tolerances

2.1 General Options

Option	Description	Default
`opmode`	Specifies the search mode used. `0:` Local search from the given nominal solution without a preceding local search (LS) `1:` Global branch-and-bound search and local search (BB+LS) `2:` Global adaptive random search and local search (GARS+LS) `3:` Global multistart random search and local search (MS+LS)	`3`

Option	Description	Default
tlimit	Time limit in seconds. This is equivalent to the GAMS option reslim. If specified, this overrides the GAMS reslim option.	1000
log_time	Iteration log time interval in seconds. Log output occurs every log_time seconds.	0.5
log_iter	Iteration log time interval. Log output occurs every log_iter iterations.	10
log_err	Iteration log error output. Error reported (if applicable) every log_err iterations.	10
debug	Debug option. Prints out complete LGO status report to listing file. 0: No 1: Yes	0
callConopt	Number of seconds given for cleanup phase using CONOPT. CONOPT terminates after at most callConopt seconds. The cleanup phase determines duals for final solution point.	5
help	Prints out all available GAMS/LGO solver options in the log and listing files.	

Note that the local search operational mode (opmode 0) is the fastest, and that it will work for convex, as well as for some non-convex models. If the model has a highly non-convex (multiextremal) structure, then at least one of the global search modes should be used. It may be a good idea to apply all three global search modes, to verify the global solution, or perhaps to find alternative good solutions. Usually, opmode 3 is the safest (and slowest), since it applies several local searches; opmodes 1 and 2 launch only a single local search from the best point found in the global search phase.

2.2 Limits and Tolerances

Option	Description	Default
g_maxfct	Maximum number of merit (model) function evaluations before termination of global search phase (BB, GARS, or MS). In the default setting, n is the number of variables and m is the number of constraints. The difficulty of global optimization models varies greatly: for difficult models, g_maxfct can be increased as deemed necessary.	500(n+m)
max_nosuc	Maximum number of merit function evaluations in global search phase (BB, GARS, or MS) where no improvement is made. Algorithm phase terminates upon reaching this limit. The default of -1 uses 100(nvars+ncons), where nvars is the number of variables and ncons the number of constraints.	100(n+m)
penmult	Constraint penalty multiplier. Global merit function is defined as objective + the constraint violations weighted by penmult.	100
acc_tr	Global search termination criterion parameter (acceptability threshold). The global search phase (BB, GARS, or MS) ends, if an overall merit function value is found in the global search phase that is not greater than acc_tr.	-1.0E+10
fct_trg	Target objective function value (partial stopping criterion in internal local search phase).	-1.0E+10
fi_tol	Local search (merit function improvement) tolerance.	1.0E-06
con_tol	Maximal constraint violation tolerance in local search.	1.0E-06
kt_tol	Kuhn-Tucker local optimality condition violation tolerance.	1.0E-06
irngs	Random number seed.	0
var_lo	Smallest (default) lower bound, unless set by user.	-1.0E+06
var_up	Largest (default) upper bound, unless set by user.	1.0E+6
bad_obj	Default value for objective function, if evaluation errors occur.	1.0E+8

Note that if model-specific information is known (more sensible target objective/merit function value, tolerances, tighter variable bounds), then such information should always be used, since it may help to solve the model far more efficiently than the usage of 'blind' defaults.

3 The GAMS/LGO Log File

The GAMS/LGO log file gives much useful information about the current solver progress and its individual phases. For illustration, we use the nonconvex model `mhw4d.gms` from the GAMS model library:

```
$Title Nonlinear Test Problem (MHW4D,SEQ=84)

$Ontext
Another popular testproblem for NLP codes.

Wright, M H, Numerical Methods for Nonlinearly Constrained Optimization.
PhD thesis, Stanford University, 1976.
$Offtext

Variables m, x1, x2, x3, x4, x5;
Equations funct, eq1, eq2, eq3;

funct.. m =e= sqr(x1-1)      + sqr(x1-x2)     + power(x2-x3,3)
            + power(x3-x4,4) + power(x4-x5,4) ;
eq1.. x1 + sqr(x2) + power(x3,3) =e= 3*sqrt(2) + 2 ;
eq2.. x2 - sqr(x3) + x4          =e= 2*sqrt(2) - 2 ;
eq3.. x1*x5 =e= 2 ;

Model wright / all / ;

x1.l = -1; x2.l = 2; x3.l = 1; x4.l = -2; x5.l = -2;
Solve wright using nlp minimizing m;
```

Note that the solution given by LGO (shown on the next page) corresponds to the global minimum. For comparison, note that local scope nonlinear solvers will not find the global solution, unless started from a suitable neighbourhood (i.e., the model- and solver-specific region of attraction) of that solution.

In this example we use an option file to print out log information every 500 iterations, regardless of the elapsed time. Note that we set the `log_time` option to 0 to ignore the `log_time` interval.

```
LGO 1.0       May  15, 2003 LNX.LG.NA 21.0 001.000.000.LXI Lib001-030502

    LGO Lipschitz Global Optimization
    (C) Pinter Consulting Services, Inc.
    129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
    E-mail : jdpinter@hfx.eastlink.ca
    Website: www.dal.ca/~jdpinter

--- Using option file C:/GAMSPROJECTS/LGODOC/LGO.OPT
    > log_iter 500
    > log_time 0

    3 defined, 0 fixed, 0 free
    6 +/- INF bound(s) have been reset
    1 LGO equations and 3 LGO variables
```

The first part prints out information about the model size after presolve. In this particular problem, the original model had 4 rows, 6 columns, and 14 non-zeroes, of which 3 were defined constraints, meaning that they could be eliminated via GAMS/LGO presolve techniques. Note that none of these were fixed or free constraints. Furthermore, LGO presolve reduced the model size further to 1 row (LGO equations) and 3 columns (LGO variables).

The main log gives information for every *n* iterations about current progress. The main fields are given in the table below:

Field	Description
Iter	Current iteration.
Objective	Current objective function value.
SumInf	Sum of constraint infeasibilities.
MaxInf	Maximum constraint infeasibility.
Seconds	Current elapsed time in seconds.
Errors	Number of errors and type. Type can either be D/E: Evaluation error B: Bound violation.

```
Iter      Objective     SumInf   MaxInf     Seconds  Errors
 500   4.515428E-01   5.76E-02  5.8E-02       0.007
1000   6.700705E-01   5.03E-05  5.0E-05       0.014
1500   2.765930E+00   6.25E-04  6.2E-04       0.020
2000   2.710653E+00   1.55E-02  1.6E-02       0.026
2500   4.016702E+00   1.44E-02  1.4E-02       0.032
3000   4.865399E+00   2.88E-04  2.9E-04       0.038
3500   4.858826E+00   3.31E-03  3.3E-03       0.044
4000   1.106472E+01   1.53E-02  1.5E-02       0.050
4500   1.595505E+01   1.56E-06  1.6E-06       0.055
5000   1.618715E+01   2.17E-05  2.2E-05       0.062
5500   1.618987E+01   3.45E-04  3.5E-04       0.067
6000   1.985940E+01   4.03E-04  4.0E-04       0.074
6500   1.624319E+01   5.64E-03  5.6E-03       0.079
7000   1.727653E+01   8.98E-05  9.0E-05       0.086
7500   1.727033E+01   3.03E-03  3.0E-03       0.091
7840   2.933167E-02   0.00E+00  0.0E+00       0.097
```

LGO then reports the termination status, in this case globally optimal, together with the solver resource time. The resource time is also disaggregated by the total time spent performing function evaluations and the number of milliseconds (ms) spent for each function evaluation.

```
--- LGO Exit: Terminated by solver - Global solution
    0.047 LGO Secs (0.015 Eval Secs, 0.001 ms/eval)
```

A local solver such as CONOPT can be called to compute marginal values. To invoke a postsolve using CONOPT, the user specifies the `callConopt` option with a positive value, indicating the number of seconds CONOPT is given to solve. See the LGO option section for further details.

Illustrative References

R. Horst and P. M. Pardalos, Editors (1995) *Handbook of Global Optimization*. Vol. 1. Kluwer Academic Publishers, Dordrecht.

P. M. Pardalos and H. E. Romeijn, Editors (2002) *Handbook of Global Optimization*. Vol. 2. Kluwer Academic Publishers, Dordrecht.

J. D. Pintér (1996) *Global Optimization in Action*, Kluwer Academic Publishers, Dordrecht.

J. D. Pintér (2001) *Computational Global Optimization in Nonlinear Systems: An Interactive Tutorial*, Lionheart Publishing, Atlanta, GA.

J. D. Pintér (2002) Global optimization: software, tests and applications. Chapter 15 (pp. 515-569) in: Pardalos and Romeijn, Editors, *Handbook of Global Optimization*. Vol. 2. Kluwer Academic Publishers, Dordrecht.