GP

GP

Introduction

Overview

Welcome to the TOMLAB /GP (Geometric Programming) User's Guide. TOMLAB /GP includes the a geometric programming solver package and a convenient interface to The MathWorks' MATLAB.

The following sections describe the algorithm and TOMLAB format in more detail. There are several test problem included with the TOMLAB distribution that illustrates the use. The following example will create and run the first test case.

```
Prob = probInit('gp_prob', 1); Result  = tomRun('GP',  Prob,  1);
```

Contents of this Manual

- #Summary provides a basic overview of the geometric programming solver.
- #Geometric Programming (GP) describes geometric programming in detail.
- #The Algorithm gives algorithmic details.
- #A GP Example illustrates how to solve a simple test case.
- #TOMLAB Test Set contains three examples to assist in the modeling of geometric programming problems.
- #Solution Output Files shows the screen and file output.

More information

Please visit the following links for more information:

- http://tomopt.com/tomlab/products/gp/ [1],

Prerequisites

In this manual we assume that the user is familiar with linear/nonlinear programming, setting up problems in TOMLAB (in particular constrained nonlinear (**con**) problems) and the Matlab language in general.

Using the Matlab Interface

The GP solver is accessed via the *tomRun* driver routine, which calls the *coplgpTL* interface routine. The solver itself is located in the MEX file *coplgp.dll*.

The interface routines.

Function	Description
coplgpTL	The interface routine called by the TOMLAB driver routine *tomRun*. This routine then calls the MEX file *gp*
gpAssign	The routine that creates a GP problem in the TOMLAB format.

GP Solver Reference

A detailed description of the TOMLAB /GP solver interface is given below. Also see the M-file help for *coplgpTL.m*.

coplgpTL

Purpose

Solve geometric programming problems as described below.

Calling Syntax

```
Prob = gpAssign( ... );
Result = tomRun('GP',Prob,...)
```

Description of Inputs

Prob Problem description structure. The following fields are used:

Input	Description
GP.A	Linear constraint matrix for dual problem.
GP.coef	Posynomial coefficient vector.
GP.nterm	Number of terms in objective and constraints.
GP.moremem	Extra memory to allocate. Default 300*n + 300*m, where n is the total number of terms and m is the number of variables. If this is not enough the interface will double the memory until it is enough and try to solve it. The final value of moremem will be returned in Result.GP.lackmem. If you solve many similar problems in a sequence, set Prob.GP.moremem to Result.GP.lackmem in order to avoid many unnecessary memory allocations.
PriLevOpt	Print level in the solver. See GP.options.PRILEV.
GP	Structure with GP solver specific fields.
PrintFile	Name of file to print progress information and results to. The output written to file is always with a print level of 2. Default: Empty, none that is.
GP.options	Structure with special fields for the GP solver:
PRILEV	Print level of GP solver. Has precedence over Prob.PriLevOpt. == 0 silent >= 1 normal output with iteration log >= 2 more verbose output >= 10 memory debug output >= 100 extreme debug output Default: 0
ITLM	Iteration limit. Default: 100
TOLX	Zero tolerance on the dual variables. Default: 1e-14
TOLZ	Zero tolerance on the constraint values. Default: 1e-14
EPSP	Feasibility tolerance on primal problem. Default: 1e-6

EPSD	Feasibility tolerance on dual problem. Default: 1e-6
EPSC	Feasibility tolerance on the linear dual constraints. Default: 1e-9
TOLPIV	Pivoting zero tolerance. Default: 1e-14
FRAC	Fractional decrease of steplength. Default: 0.9995
BIG	Value to treat as infinite. Default: 1e20

Description of Outputs

Result Structure with result from optimization. The following fields are set:

Output	Description
f_k x_k	Function value at optimum. Optimal point.
xState	State of each variable: 0/1/2/3: free / on lower bnd / on upper bnd / fixed.
ExitFlag	Exit status. The following values are used: 0: Optimal solution found. 1: Maximum number of iterations reached. 2: (Possibly) unbounded problem. 4: (Possibly) infeasible problem.
ExitText	The exit text according to Inform.
Inform	Status value returned from the GP solver. 0: Optimal: find an optimal solution. 1: Unbounded: the reformulated problem is unbounded. 2: Infeasible: the reformulated problem is infeasible. 3: Iteration limit: reach the iteration limit before desired accuracy. 4: Numerical difficulties: encounter numerical problems. 5: Primal or dual infeasible: the original problems are infeasible. 6: Insufficient space: memory space is not sufficient. 7: Data file error: find input data errors.
Solver	Name of the solver (GP).
SolverAlgorithm	Description of the solver.
GP.lackmem	See description for input Prob.GP.morememem.

Summary

COPL GP (Computational Optimization Program Library: Geometric Programming) is a highly efficient and accurate computer program developed in Computational Optimization Laboratory, Department of Management Science, The University of Iowa, Iowa City, IA 52242, USA (http://dollar.biz.ui [2] wa.edu/col/ye [3]) for solving the classical posynomial geometric programming problems. The approach is by means of a primal-dual algorithm developed simultaneously for (i), the dual geometric program after logarithmic transformation of its objective function and (ii), its Lagrangian dual program. Under rather general assumptions, the mechanism defines a primaldual infeasible path from a specially constructed, perturbed Karush-Kuhn-Tucker system. *Subfeasible*

solutions, are generated for each program whose primal and dual objective function values converge to the respective primal and dual program values.

The basic technique is one of a predictor-corrector type involving Newton's method applied to the perturbed KKT system, coupled with effective techniques for choosing iterate directions and step lengths. Sophisticated implementation techniques and advanced sparse matrix factorizations are used to take advantage of the very special structure of the Hessian matrix of the logarithmically transformed dual objective function.

Computational results on 19 of the most challenging *GP* problems found in the literature are encouraging. The performance indicates that the algorithm is effective regardless of the *degree of difficulty*, which is a generally accepted measure in geometric programming.

Geometric programming *GP* is a very broad class of mathematical problems which is useful in the study of a variety of optimization problems. Its great impact has been in the areas of **engineering design** , **economics & statistics** ,**manufacturing** and **chemical equilibrium**.

Geometric Programming (GP)

The *primal GP* is :

$$(GP) \quad V_{GP} := \begin{array}{ll} \text{minimize} & g_0(t) \\ \text{subject to} & g_k(t) \le 1, \quad k = 1, 2, \ldots, p \\ & t_i > 0, \qquad i = 1, 2, \ldots, m \end{array}$$

where

$$g_0(t) \;=\; \sum_{j=1}^{n_0} c_j t_1^{a_{1j}} \ldots t_m^{a_{mj}} g0$$

$$g_k(t) \;=\; \sum_{j=n_{k-1}+1}^{n_k} c_j t_1^{a_{1j}} \ldots t_m^{a_{mj}}, \quad k = 1, 2, \ldots, p.$$

Given exponents $a_{ij}$for the ith variable in the jth product term, $i = 1, \ldots, m$and $j = 1, \ldots, n_p$, are arbitrary real constants and term coefficients $c_j$are positive.

Here, $g_0(t)$is called the objective function and $g_k(t)$the kth constraint function, where $n_0$is the number of product terms in the objective function and $(n_k - n_{k-1})$is the number of product terms in the $k$constraint function. Therefore, the number of the total product terms is n_p, where $p$is the number of constraint functions. The vector t contains $m$variables, denoted by $t_1, \ldots, t_m$.

The program solves *posynomial GP*, and so we shall remove *posynomial* with this understanding.

The dual to *GP* is

$$(GD) \quad V_{GD} := \begin{array}{ll} \text{maximize} & \prod_{j=1}^{n_p} (c_j / x_j)^{x_j} \prod_{k=1}^{p} \lambda_k^{\lambda_k} \\ \text{subject to} & \sum_{j=1}^{n0} x_j = 1, \\ & \sum_{j=1}^{n_p} x_j a_{ij} = 0, \quad i = 1, 2, \ldots, m \\ & x_j \ge 0, \quad j = 1, 2, \ldots, n_p \end{array}$$

where

$$\lambda_k = \sum_{j=n_{k-1}+1}^{n_k} x_j, \;\; k = 1, 2, \ldots, p.$$

For a (primal) *GP* having *m* variables (t_i), *p* constraints and n_p (posynomial) product terms we see that (dual) *GD* has n_p non-negative variables (x_j) in *m* + 1 linear equations. In the literature the *degree of difficulty* of a *GP* is defined by: degree of difficulty = n_p - *m* - *1*.

Let *F* (*x*) denote the negative of the logarithm of the objective function of (*GD*), i.e.,

$$F(x) = \sum_{j=1}^{n_0} x_j \ln(\frac{x_j}{c_j}) + \sum_{k=1}^{p} \sum_{j=n_{k-1}+1}^{n_k} x_j \ln(\frac{x_j}{c_j \lambda_k}).$$

We see that (*GD*) is a linearly constrained convex programming problem,

$$\begin{array}{ll} \text{minimize} & F(x) \\ \text{subject to} & Ax = b, \quad x \geq 0, \end{array}$$

where the coefficient matrix is given by

$$A = \begin{pmatrix} 1 & ... & 1 & 0 & ... & 0 & ... & 0 & ... & 0 \\ a_{1,1} & ... & a_{1,n_0} & a_{1,n_0+1} & ... & a_{1,n_1} & ... & a_{1,n_{p-1}+1} & ... & a_{1,n_p} \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ a_{m,1} & ... & a_{m,n_0} & a_{m,n_0+1} & ... & a_{m,n_1} & ... & a_{m,n_{p-1}+1} & ... & a_{m,n_p} \end{pmatrix}$$

and right side hand

$$b^T = (1, 0, ..., 0) \in R^{m+1}.$$

Note that the first n_0 entries of the first row of *A* are ones, and data a_{ij} are stored from row 2 to row *m* + 1 in *A*.

The Algorithm

The program is an implementation of an interior-point algorithm for linear constrained convex programming which when applied to geometric programming solves both primal and dual *GP* problems simultaneously. Using the perturbed KKT system of the dual *GP* , a central-path is defined which converges to a solution of both the primal and dual *GP* programs. The algorithm does not require the existence of an interior point for either program. Moreover, the algorithm has the feature of generating *subfeasible solutions*, when the primal geometric program is inconsistent but subconsistent. Computing functional values of subfeasible solutions gives rise to more conceivable duality states for the primal-dual pair. In this manual we limit these possibilities by considering "boundedly subfeasible" sequences for which our algorithm delivers a path whose primal-dual gap can be made arbitrarily small, even for the case where one of the pair of dual problem is inconsistent. **In addition, the objective functions need not be differentiable at an optimal solution of a given consistent program.**

Each iteration of the algorithm computes the Newton direction from a perturbed KKT system involving two parameters, $\gamma$and η. The affine direction (corresponding to a particular setting $\gamma = 0, \eta = 1$) is used to predict reductions in both the feasibility and complementarity residuals. Then proper parameters $\gamma, \eta$are chosen for the KKT system which is solved for the direction again. The indefinite reduced KKT matrix is factorized by performing diagonal pivots whose orders are chosen in terms of minimum degree to maintain the sparsity of the factors. A static data structure associated with the chosen pivoting order is allocated by a single symbolic factorization and used in subsequent iterations. Tactics for taking large steps and reset dual slack variables according to certain rules are implemented.

Computational results indicate that the algorithm, coupled with these new techniques, leads to an efficient and stable implementation for solving primal and dual geometric programs simultaneously. The results indicate that the standard geometric programming measure of difficulty is no barrier to the methods.

A GP Example

This example is called Dembo78:

$$(\mathrm{DP})\ \min\{t_1t_2 + t_1^{-1}t_2^{-1} | (1/4)t_1^{1/2} + t_2 \leq 1,\ t_1 > 0,\ t_2 > 0\}.$$

The *GP* dual to Dembo78 has a unique optimal solution, $x = (0.5, 0.5, 0, 0)$:

$$\begin{array}{lll} (DD) \quad \max & (1/x_1)^{x_1}(1/x_2)^{x_2}(0.25/x_3)^{x_3}(1/x_4)^{x_4}(x_3+x_4)^{(x_3+x_4)} & \\ \text{subject to} & x_1 + x_2 & = 1 \\ & x_1 - x_2 + 0.5x_3 & = 0 \\ & x_1 - x_2 + x_4 & = 0 \\ & x \geq 0 & \end{array}$$

The TOMLAB commands are:

```
Prob = gpAssign([2; 2], [1 1 1/4 1],...
       [1 -1 0.5 0; 1 -1 0 1]');

Result = tomRun('gp', Prob, 1);
```

The problem is also defined in *gp_prob*:

```
Prob = probInit('gp_prob', 1);
Result  = tomRun('gp', Prob,  1);
```

TOMLAB Test Set

There are several test problems included with the TOMLAB distribution. Three of the problems are illustrated below (P1, P4 and P10). These problems are created and solved by executing:

```
Prob1 = probInit('gp_prob', 3);   % Create P1.
Prob2 = probInit('gp_prob', 6);   % Create P4.
Prob3 = probInit('gp_prob', 12);  % Create P10A.

Result1 = tomRun('GP', Prob1, 2);   % Solve P1.
Result2 = tomRun('GP', Prob2, ,2);  % Solve P4.
Result3 = tomRun('GP', Prob3, ,2);  % Solve P10A.
```

Problem P1

$$\begin{array}{lll} (P1) \quad \min & 5x_1 + 50000x_1^{-1} + 20x_2 + 72000x_2^{-1} + 10x_3 + 144000x_3^{-1} & \\ \text{subject to} & 4x_1^{-1} + 32x_2^{-1} + 120x_3^{-1} <= 1 & \\ & x \geq 0 & \end{array}$$

Using TOMLAB the problem is modeled as:

Problem P1 in TOMLAB

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \end{pmatrix}^T$$

$$coef = \begin{pmatrix} 5 & 50000 & 20 & 72000 & 10 & 144000 & 4 & 32 & 120 \end{pmatrix}^T$$

$$nterm = \begin{pmatrix} 6 & 3 \end{pmatrix}^T$$

```
Prob = gpAssign(nterm, coef,  A,  'P1');
Result  = tomRun('GP',  Prob,  ,2);
```

similarly for the other test cases:

Problem P4

$$\begin{aligned} (P1) \quad \min \quad & x_1^{-1}x_2^{-1}x_3^{-1} \\ \text{subject to} \quad & 2x_1 + x_2 + 3x_3 <= 1 \\ & x_1 + 3x_2 + 2x_3 <= 1 \\ & x_1 + x_2 + x_3 <= 1 \\ & x \geq 0 \end{aligned}$$

Problem P4 in TOMLAB

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}^T$$

$$coef = \begin{pmatrix} 1 & 2 & 1 & 3 & 1 & 3 & 2 & 1 & 1 & 1 \end{pmatrix}^T$$

```
Prob = gpAssign(nterm, coef,  A,  'P4'); Result  = tomRun('GP',  Prob,  ,2);
```

Problem P10A

$$
\begin{aligned}
(P1) \quad \min \quad & 2x_1^{0.9}x_2^{-1.5}x_3^{-3} + 5x_4^{-0.3}x_5^{2.6} + 4.7x_6^{-1.8}x_7^{-0.5}x_8 \\
\text{subject to} \quad & 7.2x_1^{-3.8}x_2^{2.2}x_3^{4.3} + 0.5x_4^{-0.7}x_5^{-1.6} + 0.2x_6^{4.3}x_7^{-1.9}x_8^{8.5} <= 1 \\
& 10x_1^{2.3}x_2^{1.7}x_3^{4.5} <= 1 \\
& 0.6x_4^{-2.1}x_5^{0.4} <= 1 \\
& 6.2x_6^{4.5}x_7^{-2.7}x_8^{-0.6} <= 1 \\
& 3.1x_1^{1.6}x_2^{0.4}x_3^{-3.8} <= 1 \\
& 3.7x_4^{5.4}x_5^{1.3} <= 1 \\
& 0.3x_6^{-1.1}x_7^{7.3}x_8^{-5.6} <= 1 \\
& x \geq 0
\end{aligned}
$$

Problem P10A in TOMLAB

$$
A = \begin{pmatrix}
0.9 & 0 & 0 & -3.8 & 0 & 0 & 2.3 & 0 & 0 & 1.6 \\
0 & 0 & & & & & & & & \\
-1.5 & 0 & 0 & 2.2 & 0 & 0 & 1.7 & 0 & 0 & 0.4 \\
0 & 0 & & & & & & & & \\
-3 & 0 & 0 & 4.3 & 0 & 0 & 4.5 & -2.1 & 0 & -3.8 \\
0 & 0 & & & & & & & & \\
0 & -0.3 & 0 & 0 & -0.7 & 0 & 0 & -2.1 & 0 & 0 \\
5.4 & 0 & & & & & & & & \\
0 & 2.6 & 0 & 0 & -1.6 & 0 & 0 & 0.4 & 0 & 0 \\
1.3 & 0 & & & & & & & & \\
0 & 0 & -1.8 & 0 & 0 & 4.3 & 0 & 0 & 4.5 & 0 \\
0 & -1.1 & & & & & & & & \\
0 & 0 & -0.5 & 0 & 0 & -1.9 & 0 & 0 & -2.7 & 0 \\
0 & 7.3 & & & & & & & & \\
0 & 0 & 1 & 0 & 0 & 8.5 & 0 & 0 & -0.6 & 0 \\
0 & -5.6 & & & & & & & &
\end{pmatrix}^T
$$

$$
coef = \begin{pmatrix}
2 & 5 & 4.7 & 7.2 & 0.5 & 0.2 & 10 & 0.6 & 6.2 & 3.1 \\
3.7 & 0.3 & & & & & & & &
\end{pmatrix}^T
$$

```
Prob = gpAssign(nterm, coef,  A,  'P10A');
Result  = tomRun('GP',  Prob,  ,2);
```

Solution Output Files

The output file contains three sections.

- The first section "geometric programming model" describes some problem characteristics, such as the number of variables, etc.
- The second section "path-following solver for reformulated model" contains the running information during the iterative process. The primal and dual objective values represent the reformulated primal and dual problems.
- The third section "solutions for original GP" contains the optimal values of the variables, the constraint functions, the primal and dual objective functions, and the feasibility residues of the original GP problem pair.

The termination code 0 - -7 represents the following:

```
0=optimal: find an optimal solution;
1=unbounded: the reformulated problem is unbounded;
2=infeasible: the reformulated problem is infeasible;
3=iteration limit: reach the iteration limit before desired accuracy;
4=numerical difficulties: encounter numerical problems
5=primal or dual infeasible: the original problems are infeasible;
6=insufficient space: memory space is not sufficient;
7=data file error: find input data errors.
```

The last paragraph lists the CPU time: read in data, solve the problem and the sum of the two. The following is the output file for Dembo68.

```
                              =================
                              COPL_GP 1.1  2005
                              =================
              geometric programming optimizer


  geometric programming model
  ---------------------------
  primal gp vars . 2
  primal gp consts 1
  primal gp terms  4
  degree of diffic 1

  path-following solver for reformulated model
  --------------------------------------------

*- iter =   0        dual   value         =-2.00000000000000E+00
                     primal value         =-4.64038529823796E-17
    mu  = 9.233E-01  Dual   infeasibility = 3.608E-01
                     Primal infeasibility = 5.000E-01

*- iter =   1        dual   value         =-9.99533217451737E-01
                     primal value         =-3.62989378647483E-01
    mu  = 1.591E-01  Dual   infeasibility = 0.000E+00
                     Primal infeasibility = 1.872E-17
```

```
*- iter =   2        dual   value         =-7.13113168002422E-01
                     primal value         =-5.88706966976516E-01
    mu  = 3.544E-02  Dual   infeasibility = 2.294E-02
                     Primal infeasibility = 8.301E-17

*- iter =   3        dual   value         =-6.93636390346386E-01
                     primal value         =-6.89738422619333E-01
    mu  = 1.854E-03  Dual   infeasibility = 4.382E-03
                     Primal infeasibility = 2.975E-17

*- iter =   4        dual   value         =-6.93130105030368E-01
                     primal value         =-6.93115339650404E-01
    mu  = 9.392E-06  Dual   infeasibility = 2.847E-05
                     Primal infeasibility = 2.383E-17

*- iter =   5        dual   value         =-6.93147171812859E-01
                     primal value         =-6.93147164639831E-01
    mu  = 4.710E-09  Dual   infeasibility = 1.456E-08
                     Primal infeasibility = 4.592E-17

*- iter =   6        dual   value         =-6.93147180555572E-01
                     primal value         =-6.93147180551985E-01
    mu  = 2.355E-12  Dual   infeasibility = 7.282E-12
                     Primal infeasibility = 8.860E-17

*- iter =   7        dual   value         =-6.93147180559943E-01
                     primal value         =-6.93147180559941E-01
    mu  = 1.210E-15  Dual   infeasibility = 3.705E-15
                     Primal infeasibility = 5.133E-17

 -- exit : optimal solution obtained. iter 7

          dual objective value =-6.93147180559943E-01
          primal               =-6.93147180559941E-01
          Dual infeasibility   = 3.705E-15
          Primal               = 5.133E-17

  solutions for original GP
  -------------------------

 primal GP solution

        i              t(i)
        1  .421974819510E+01
        2  .236980965158E+00

 primal GP constraint value
```

```
       i   constraint value
       1  .750531607446E+00

primal GP obj value (from direct comp)      2.00000000000000E+00
primal GP infeasibility                     0.000E+00

dual   GP obj value (from direct comp)      1.99999999999999E+00
dual   GP infeasibility                     5.133E-17

 all done, termination code=    0
 0=optimal, 1=unbounded, 2=infeasible
 3=iteration limit, 4=numerical difficulties
 5=primal or dual infeasible, 6=insufficient space
 7=data file error

----- time for readin =        .02
              solving =        .01
total time in seconds =        .03
```

Tolerances for primal and dual feasibilities and complementarity are specified as

$$\epsilon_P = 10^{-8}, \quad \epsilon_D = 10^{-8}, \quad \text{and } \epsilon_C = 10^{-12}.$$

Extensive computational tests indicate that the requirement of $\epsilon_C = 10^{-12}$is very strong. The solution stopping by this specification usually is very accurate

References

[1] http://tomopt.com/tomlab/products/gp/
[2] http://dollar.biz.uiowa.edu/col/ye)
[3] http://dollar.biz.uiowa.edu/col/ye

Article Sources and Contributors

GP *Source*: http://tomwiki.com/index.php?oldid=2411 *Contributors*: Elias