

User's Guide of COPL_GP with Python

September 29, 2017

1 Introduction

Geometric Programming (GP) is a very broad class of mathematical problems which is useful in many areas such as engineering design : [8],[2], [11], and [12]; economics & statistics: [5], [17], [4], [13],[1], and [3]; manufacturing: [14], [10],[18]; chemical equilibrium: [8], [2]. COPL_GP (Computational Optimization Programming Library: Geometric Programming) is a highly efficient and accurate computer program for solving classical posynomial geometric programming problems. In particular, the program is an implementation of an interior-point algorithm for linear constrained convex programming which when applied to geometric programming solved both primal and dual GP problems simultaneously. The 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. The technical details of COPL_GP can be found in Kortanek, Xu and Ye [15]. COPL_GP was first coded in Fortran 77, and the corresponding executable code (Linux or HP) can be downloaded from <https://web.stanford.edu/~yyye/Col.html>. **This is the python version of COPL_GP, which is developed and maintained by SHUFE solver team.**

2 What is Geometric Programming (GP)

A standard (GP) is given by

$$(GP) \quad V_{GP} := \min \quad g_0(t) \\ \text{s.t.} \quad g_k(t) \leq 1 \quad k = 1, 2, \dots, p \\ t_i > 0 \quad i = 1, 2, \dots, m \quad (1)$$

where

$$g_0(t) = \sum_{j=1}^{n_0} c_j t_1^{a_{1j}} \cdots t_m^{a_{mj}} \quad (2)$$

is the objective function with n_0 the number of product terms, and

$$g_k(t) = \sum_{j=n_{k-1}+1}^{n_k} c_j t_1^{a_{1j}} \cdots t_m^{a_{mj}} \quad (3)$$

is the k -th constraint with $n_k - n_{k-1}$ the number of product terms.

Then we define the coefficient matrix

$$A = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ a_{1,1} & \cdots & a_{1,n_0} & a_{1,n_0+1} & \cdots & a_{1,n_1} & \cdots & a_{1,n_{p-1}+1} & \cdots & a_{1,n_p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & \cdots & a_{m,n_0} & a_{m,n_0+1} & \cdots & a_{m,n_1} & \cdots & a_{m,n_{p-1}+1} & \cdots & a_{m,n_p} \end{pmatrix} \quad (4)$$

3 Call COPL_GP in Python

Step one: Installation **gfortran**, which can be downloaded from

<https://gcc.gnu.org/wiki/GFortranBinaries>;

Step two: In order to compile python code, run makefile as follows:

```
make all
```

To remove the .so file, simply run:

```
make clean
```

Step three: Enter python

```
python
```

Step four: Import function copl_gp

```
import copl_gp
```

Then we can see the format of wrapped function.

```
print(copl_gp.copl_gp.__doc__)
```

Step five: Call COPL_GP

```
copl_gp.copl_gp(m,n,nz,iat,jat,ant,clg,ncum,[ip])
```

3.1 Parameters in copl_gp

There are 8 parameters in copl_gp: m,n,nz,iat,jat,ant,clg,ncum,[ip],gpvalue,solution. We explain each of them as follows.

- “m” represents the number of rows in matrix A of (4).
- “n” represents the number of columns in matrix A .
- “nz” represents the number of nonzero entries in matrix A .
- Each integer in “iat” represents the number of nonzero entries in each column of matrix A from column 1 to column n_p . Note that n_p is the number of total terms in the GP problem or the number of rows in matrix A .
- “jat” contains all the row indices of all nonzero entries in matrix A .
- The real numbers in ‘ant’ are those nonzero entries, corresponding to the row index and column index in “jat” and “iat”.
- “clg” contains positive real numbers $c_j, j = 1, \dots, n_p$.
- The first integer in “ncum” is the number of product terms in the objective function $g_0(t)$. Then, following integers is the numbers of product terms in each of the constraint functions $g_k(t), k = 1, \dots, p$, accordingly.
- “ip” is the number of constraints in the GP problem.

There are two returns from copl_gp:

- “gpvalue” is the primal obj value in the GP problem.
- “solution” is the primal GP solution.

4 Two GP Examples

4.1 The First Example

This example is called Dembo78, [7]:

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

The A matrix is of the form:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0.5 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} \quad (5)$$

The parameters in `copl_gp` are specified as:

- `m=3`
- `n=4`
- `nz=8`
- `iat(n)=[3,3,1,1]`
- `jat(nz)=[1,2,3,1,2,3,2,3]`
- `ant(nz)=[1.0,1.0,1.0,1.0,-1.0,-1.0,0.5,1.0]`
- `clg(n)=[1.0,1.0,0.25,1.0]`
- `ip=1`
- `ncum(ip+1)=[2,2]`

The codes to solve this problem are presented below.

```
In [1]: import copl_gp
import numpy as np

In [2]: m=3;n=4;nz=8
ant=[1.0,1.0,1.0,1.0,-1.0,-1.0,0.5,1.0]
iat=[3,3,1,1]
jat=[1,2,3,1,2,3,2,3]
clg=[1.0,1.0,0.25,1.0]
ip=1
ncum=[2,2]

In [3]: gpvalue, solution = copl_gp.copl_gp(m,n,nz,iat,jat,ant,clg,ncum,ip)

In [4]: print(gpvalue)

2.0

In [5]: print(solution)

[ 4.21974819  0.23698097]
```

The following returned message summarizes the optimization progress.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```
=====
COPL_GP 1.1 1995
=====
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      = -2.000000000000000E+00
                    primal value     = 0.000000000000000E+00
    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 = 2.247E-16

*- iter = 2          dual value      = -7.13113174697980E-01
                    primal value     = -5.88706962224864E-01
    mu = 3.544E-02   Dual infeasibility = 2.294E-02
                    Primal infeasibility = 5.929E-17

*- iter = 3          dual value      = -6.93636390013630E-01
                    primal value     = -6.89738421831943E-01
    mu = 1.854E-03   Dual infeasibility = 4.382E-03
                    Primal infeasibility = 2.759E-17

*- iter = 4          dual value      = -6.93130105036837E-01
                    primal value     = -6.93115339560648E-01
    mu = 9.392E-06   Dual infeasibility = 2.847E-05
                    Primal infeasibility = 5.030E-17

*- iter = 5          dual value      = -6.93147171812459E-01
                    primal value     = -6.93147164639042E-01
    mu = 4.710E-09   Dual infeasibility = 1.456E-08
                    Primal infeasibility = 1.994E-17

*- iter = 6          dual value      = -6.93147180555571E-01
                    primal value     = -6.93147180551985E-01
    mu = 2.355E-12   Dual infeasibility = 7.282E-12
                    Primal infeasibility = 8.845E-17

--- exit : optimal solution obtained. iter          6

dual objective value = -6.93147180555571E-01
primal                = -6.93147180551985E-01
Dual infeasibility    = 7.282E-12
Primal                = 8.845E-17
```

```

58 |
59 | solutions for original GP
60 | _____
61 |
62 | primal GP solution
63 |
64 |      i                t(i)
65 |      1 0.421974819454E+01
66 |      2 0.236980965189E+00
67 |
68 | primal GP constraint value
69 |
70 |      i  constraint value
71 |      1 0.750531607442E+00
72 |
73 | primal GP obj value (from direct comp)      2.000000000000000E+00
74 | primal GP infeasibility                      0.000E+00
75 |
76 | dual   GP obj value (from direct comp)      1.99999999998408E+00
77 | dual   GP infeasibility                      8.845E-17
78 |
79 | all done, termination code=    0
80 | 0=optimal, 1=unbounded, 2=infeasible
81 | 3=iteration limit, 4=numerical difficulties
82 | 5=primal or dual infeasible, 6=insufficient space
83 | 7=data file error
84 |
85 | —— time for readin =          0.00
86 |                solving =          0.00
87 | total time in seconds =          0.00

```

4.2 The Second Example

By imposing an additional equality constraint, we obtain the second example:

$$\min \{t_1 t_2 + t_1^{-1} t_2^{-1} \mid (1/4)t_1^{1/2} + t_2 \leq 1, t_1 = 1, t_1 > 0, t_2 > 0\},$$

which can be rewritten as

$$\min \{t_1 t_2 + t_1^{-1} t_2^{-1} \mid (1/4)t_1^{1/2} + t_2 \leq 1, t_1 \leq 1, \frac{1}{t_1} \leq 1, t_1 > 0, t_2 > 0\},$$

Then the A matrix is of the form:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0.5 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (6)$$

The parameters in `copl_gp` are specified as:

- `m=3`
- `n=6`
- `nz=10`
- `iat(n)=[3,3,1,1,1,1]`
- `jat(nz)=[1,2,3,1,2,3,2,2,2]`
- `ant(nz)=[1.0,1.0,1.0,1.0,-1.0,-1.0,0.5,1.0,1.0,-1.0]`

- $clg(n)=[1.0,1.0,0.25,1.0,1.0,1.0]$
- $ip=3$
- $ncum(ip+1)=[2,2,1,1]$

The codes to solve this problem are presented below.

```
In [1]: import copl_gp
import numpy as np

In [2]: m=3;n=6;nz=10
ant=[1.0,1.0,1.0,1.0,-1.0,-1.0,0.5,1.0,1.0,-1.0]
iat=[3,3,1,1,1,1]
jat=[1,2,3,1,2,3,2,3,2,2]
clg=[1.0,1.0,0.25,1.0,1.0,1.0]
ip=3
ncum=[2,2,1,1]

In [3]: gpvalue, solution = copl_gp.copl_gp(m,n,nz,iat,jat,ant,clg,ncum,ip)

In [4]: print(gpvalue)

2.083333333337629

In [5]: print(solution)

[ 1.    0.75]
```

The following returned message summarizes the optimization progress.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```

```

=====
COPL_GP 1.1 1995
=====
geometric programming optimizer

geometric programming model
-----
primal gp vars .      2
primal gp consts     3
primal gp terms      6
degree of diffic     3

path-following solver for reformulated model
-----
*- iter = 0          dual value      = -2.000000000000000E+00
                    primal value     = 0.000000000000000E+00
    mu  = 9.489E-01  Dual infeasibility = 5.518E-01
                    Primal infeasibility = 3.571E-01

*- iter = 1          dual value      = -1.18147432303578E+00
                    primal value     = -6.19500736546381E-01
    mu  = 1.502E-01  Dual infeasibility = 7.484E-02
                    Primal infeasibility = 8.652E-02

*- iter = 2          dual value      = -8.51274563853049E-01
```

```

28      primal value      = -7.20302211404315E-01
29      mu = 3.231E-02   Dual   infeasibility = 2.138E-01
30      Primal infeasibility = 4.908E-03
31
32  *-- iter = 3         dual   value      = -7.77475771603571E-01
33      primal value      = -7.29906197498383E-01
34      mu = 8.901E-03   Dual   infeasibility = 2.612E-03
35      Primal infeasibility = 1.287E-03
36
37  *-- iter = 4         dual   value      = -7.41829230451529E-01
38      primal value      = -7.33619272030830E-01
39      mu = 1.748E-03   Dual   infeasibility = 4.244E-03
40      Primal infeasibility = 1.655E-04
41
42  *-- iter = 5         dual   value      = -7.34826721443374E-01
43      primal value      = -7.33962565606459E-01
44      mu = 2.087E-04   Dual   infeasibility = 8.640E-04
45      Primal infeasibility = 1.385E-05
46
47  *-- iter = 6         dual   value      = -7.33988411027814E-01
48      primal value      = -7.33969184629826E-01
49      mu = 5.556E-06   Dual   infeasibility = 3.208E-05
50      Primal infeasibility = 2.489E-07
51
52  *-- iter = 7         dual   value      = -7.33969201749646E-01
53      primal value      = -7.33969175099293E-01
54      mu = 8.332E-09   Dual   infeasibility = 5.339E-08
55      Primal infeasibility = 3.160E-10
56
57  *-- iter = 8         dual   value      = -7.33969175093555E-01
58      primal value      = -7.33969175080210E-01
59      mu = 4.173E-12   Dual   infeasibility = 2.675E-11
60      Primal infeasibility = 1.581E-13
61
62  — exit : optimal solution obtained. iter      8
63
64      dual objective value = -7.33969175093555E-01
65      primal                = -7.33969175080210E-01
66      Dual infeasibility    = 2.675E-11
67      Primal                = 1.581E-13
68
69  solutions for original GP
70  _____
71
72  primal GP solution
73
74      i          t(i)
75      1 0.100000000000E+01
76      2 0.749999999945E+00
77
78  primal GP constraint value
79
80      i   constraint value
81      1 0.999999999945E+00
82      2 0.100000000000E+01
83      3 0.100000000000E+01
84
85  primal GP obj value (from direct comp)      2.08333333337629E+00

```

```

86 primal GP infeasibility                2.722E-13
87
88 dual   GP obj value (from direct comp)  2.083333333333335E+00
89 dual   GP infeasibility                1.581E-13
90
91 all done, termination code=    0
92 0=optimal, 1=unbounded, 2=infeasible
93 3=iteration limit, 4=numerical difficulties
94 5=primal or dual infeasible, 6=insufficient space
95 7=data file error
96
97 ——— time for readin =          0.00
98                solving =          0.00
99 total time in seconds =          0.00

```

References

- [1] H. El Barmi, and R.L. Dykstra, "Restricted multinomial maximum likelihood estimation based upon Fenchel duality," *Statistics and Probability Letters* 21 (1994) 121-130.
- [2] C.S. Beightler and D.T. Phillips, *Applied Geometric Programming* (John Wiley & Sons, NY, 1976).
- [3] D.L. Bricker, K.O. Kortanek and L. Xu, "Maximum likelihood estimates with order restrictions on probabilities and odds ratios: a geometric programming approach", *Applied Mathematical and Computational Sciences*, The University of IA, Iowa City, Iowa 52242, September, 1995
- [4] A. Charnes, W.W. Cooper, B. Golany and J. Masters, "Optimal design modification by geometric programming and constrained stochastic network models," *International Journal on Systems Science* 19 (1988) 825-844.
- [5] A. Charnes, W.W. Cooper and K.O. Kortanek, "Semi-infinite programming, differentiability and geometric Programming," *Journal of Mathematical Sciences* 6 (1971) 19-40.
- [6] R.S. Dembo, "A set of geometric programming test problems and their solutions," *Mathematical Programming* 10 (1976) 192-213.
- [7] R.S. Dembo, "Dual to primal conversion in geometric programming," *Journal of Optimization Theory and Applications* 26 (1978) 243-252.
- [8] R.J. Duffin, E.L. Peterson and C. Zener, *Geometric Programming—Theory and Application* (John Wiley & Sons, NY, 1967).
- [9] R.J. Duffin, "Infinite Programs", in: H. W. Kuhn and A. W. Tucker, eds., *Linear Inequalities and Related Systems*, (Princeton University Press, Princeton, NJ, 1956) pp.157-170
- [10] J. Dupacová, P. Charamza and J. Mádl "On stochastic aspects of a metal cutting problem," Department of Statistics, Charles University, Sokolovská 83, 18600 Prague, Czech Republic, June, 1993.
- [11] A.V. Fiacco and A. Ghaemi, "Sensitivity analysis of a nonlinear water pollution control model using an upper hudson river data base," The George Washington University, Washington, D.C., August, 1981.
- [12] A.V. Fiacco and A. Ghaemi, "Sensitivity and parametric bound analysis of an electric power generator GP model: optimal steam turbine-exhaust annulus and condenser sizes, serial T-437," The George Washington University, Washington, D.C., August, 1981
- [13] R. Jagannathan, "A stochastic geometric programming problem with multiplicative recourse," *Operations Research Letters* 9 (1990) 99-104.
- [14] S. Jha, K.O. Kortanek and H. No, "Lotsizing and setup time reduction under stochastic demand: a geometric programming approach," Working paper series No. 88-12, College of Business Administration, The University of Iowa, Iowa City, June, 1988

- [15] K.O. Kortanek, X. Xu and Y. Ye, "An infeasible interior-point algorithm for solving primal and dual geometric programs," *Mathematical Programming* 76 (1996) 155-181.
- [16] M. J. Rijckaert and X. M. Martens, "Bibliographical note on geometric programming," *Journal of Optimization Theory and Applications* 26 (1978) 325-337
- [17] T. Robertson, F.T. Wright and R.L. Dykstra, *Order Restricted Statistical Inference* (John Wiley and Sons, NY, 1988).
- [18] C.H. Scott and T.R. Jefferson, "Allocation of resources in project management," *International Journal on Systems Science* 26 (1995) 413-420.