Algorithm 728: FORTRAN Subroutines for Generating Quadratic Bilevel Programming Test Problems

PAUL H. CALAMAI
University of Waterloo
and
LUIS N. VICENTE
Universidade de Coimbra

This paper describes software for generating test problems for quadratic bilevel programming. The algorithm constructs problems with a number of favorable properties that can be selected and controlled by the user. The intention is to provide a set of FORTRAN 77 routines that can be used for testing, verifying, and comparing solution techniques for these problems.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization—*nonlinear programming*; G.4 [**Mathematics of Computing**]: Mathematical Software—*certification and testing*; *efficiency*.

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Bilevel programming, FORTRAN, quadratic separable programs, test problems

1. INTRODUCTION

In Calamai and Vicente [1994] (this issue) we describe a technique for generating random quadratic bilevel programming problems. To date, the authors know of no other technique that fulfills the same purpose. Our hope is that these codes will be used for testing, verifying, and comparing new (and established) solution methods.

This paper was completed while P. H. Calamai was on a research sabbatical at the Universidade de Coimbra, Portugal. Support of this work has been provided by the Instituto Nacional de Investigação Científica de Portugal (INIC) under Contract 89/EXA/5 and by the Natural Sciences and Engineering Research Council of Canada Operating Grant 5671
Authors' addresses: P. H. Calamai, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; and L. N. Vicente, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77251.

To encourage the use of these codes, every effort has been made to keep them as flexible as possible. At the same time, the code has been designed to meet the requirements of several existing solution techniques.

2. IMPLEMENTATION AND DESIGN

This code is implemented entirely in portable ANSI standard FORTRAN 77 with one exception: Both uppercase and lowercase letters are used to improve readability. We also use some level-1 BLAS [Lawson et al. 1979a; 1979b] routines (obtained from the Linpack project [Dongarra et al. 1978]) to provide modularity and portability.

There are several routines in the suite:

(1) *Subroutine qbpgen.* This is the central routine of the suite and the one that controls the properties of the problems that are generated. See Section 3.
(2) *Subroutine qbpfun.* This routine evaluates the functions and constraints generated by *subroutine qbpgen.* See Section 3.
(3) *Subroutine qbpchk.* This service routine checks a candidate solution against all minimizers of the generated problem. See Section 3.
(4) *Subroutine qbpax.* This routine is used to evaluate the left-hand side of the constraints.
(5) *Subroutine qbpmx.* This routine applies the transformation constructed by *subroutine qbpmkm.*
(6) *Subroutine qbpsx.* This routine is used by *subroutine qbpfun* to evaluate the lower-level objective of the generated problems.
(7) *Subroutine qbpmkm.* This routine constructs the transformation.
(8) *Subroutine qbpdot.* This routine splits the inner product of two (potentially) sparse vectors between two scalars.
(9) *Subroutine qbpflp.* This routine computes a vector minus a constant times a vector.
(10) *Subroutine dload.* This routine copies a scalar into a vector.
(11) *Function rand.* This portable random number generator is completely described in Schrage [1979].
(12) *Modified level-1 BLAS routines.* These routines have been renamed to axpy, copy, dot, scal, and nrm2. See Lawson et al. [1979a; 1979b].

3. USAGE NOTES

Each test problem that is generated (i.e., each call to *subroutine qbpgen)* has properties that depend on the choice of values given to the following parameters passed to *subroutine qbpgen*:

—the number of upper-level and lower-level variables, nx and ny;
—the solution characteristics, $m1$ through $m4$, and the vectors $rhom2$ and $rhom4$; and
—the characteristics of the transformation, $lcondm$ and $non0$.

The influence of these parameters is described in the code comments and in Calamai and Vicente [1994]. Example calls to *subroutine qbpgen* appear in *program usage*, which accompanies this code. An annotated description of these examples can be obtained by stripping out the comments in this program.

Two additional subroutines that may be called by the user are *subroutine qbpfun* and *subroutine qbpchk*. Such calls are demonstrated in *program usage*. A call to *subroutine qbpfun*

subroutine qbpfun(v, upper, lower, res)

evaluates the quadratic bilevel objectives (scalar parameters *upper* and *lower*) and the constraint residuals (vector parameter *res*) at the point passed (vector parameter v). A call to *subroutine qbpchk*

subroutine qbpchk(v, mr, errl2, type)

compares the candidate solution (passed in vector parameter v) against all minimizers. It is important to note that this check is made in the untransformed space (to do so in the transformed space would be a combinatorial process) using the point wv (the point in the untransformed space corresponding to v). On termination, *qbpchk* returns the vector mr (the point in the transformed space corresponding to r, where r is the minimizer closest to wv), the Euclidean distance between wv and r (in scalar parameter $errl2$), and the classification of the minimizer mr (in scalar parameter *type*).

4. INSTALLATION NOTES

4.1 Preliminaries

The precision of the programs in this suite can be changed in one of two ways. The easiest approach is to obtain the FORTRAN 77 program *change.f* (see Grcar [1992]) and to use it to manipulate the precision *change blocks* in all routines in the suite. This procedure is described in Grcar [1992]. Alternatively, a text editor can be used to make a single-precision version of the suite by appending a FORTRAN comment character to the start of all lines between the precision double *change blocks* (i.e., those lines between sequential **precision > double** and **end precision > double** comments) and deleting the FORTRAN comment character at the start of all lines between the precision single *change blocks* (i.e., those lines between sequential **precision > single** and **end precision > single** comments). A double-precision version of this suite is obtained, in this fashion, by interchanging the append and delete operations in the above instructions.

4.2 Linking

This suite makes extensive use of some level-1 BLAS routines [Lawson et al. 1979a; 1979b]. A modified version of these routines, which incorporates the precision *change blocks* described above, has been included with the suite. Users may wish to strip these modified routines from the suite, modify the

calls to the *modified* BLAS routines to conform with the appropriate (i.e., correct-precision) BLAS calls, and to link to the appropriate BLAS library.

4.3 Verification

A rudimentary test of the installation can be performed by compiling *program verify* which is included with the suite, linking to the suite, and running the resulting module.

4.4 Limitations

The parameter *maxn* controls the size of the largest problem that can be generated by the suite. The value of *maxn* is currently fixed at 2000 in a parameter statement in subroutines *qbpgen*, *qbpfun*, *qbpchk*, *qbpmx*, and *qbpmkm* and in program *verify*. A user wishing to generate larger problems must change these parameter statements accordingly.

REFERENCES

CALAMAI, P. H., AND VICENTE, L. N. 1994. Generating quadratic bilevel programming test problems. *ACM Trans. Math. Softw. 20*, 1, 119–135.

DONGARRA, J. J., BUNCH, J. R., MOLER, C. B., AND STEWART, G. W. 1978. *LINPACK Users Guide*. SIAM Publications, Philadelphia, Pa.

GRCAR, J. F. 1992. The change tool for changing programs and scripts. Sandia Rep. SAND92-8225, UC-405, Sept. Sandia National Labs., Albuquerque, New Mex.

LAWSON, C. L., HANSON, R. J., KINCAID, D. R., AND KROGH, F. T. 1979a. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Softw. 5*, 3 (Sept.), 308–323.

LAWSON, C. L., HANSON, R. J., KINCAID, D. R., AND KROGH, F. T. 1979b. Algorithm 539: Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Softw. 5*, 3 (Sept.), 324–325.

SCHRAGE, L. 1979. A more portable Fortran random number generator. *ACM Trans. Math. Softw. 5*, 2 (June), 132–138.

Received December 1991; revised October 1992; accepted April 1993