Algorithm 774: Fortran Subroutines for Generating Box-Constrained Optimization Problems

FRANCISCO FACCHINEI
Università di Roma “La Sapienza”
and
JOAQUIM JÚDICE and JOÃO SOARES
Universidade de Coimbra

We describe a set of Fortran routines for generating box-constrained nonlinear programming test problems. The technique, as described by Facchinei et al. (this issue), allows the user to control relevant properties of the generated problems.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran 77*; G.1.6 [**Numerical Analysis**]: Optimization; G.4 [**Mathematics of Computing**]: Mathematical Software—*algorithm analysis*; *certification and testing*

General Terms: Algorithms, Performance, Verification

Additional Key Words and Phrases: Nonlinear programming test problems, optimization, test problems generation

1. INTRODUCTION

We present a set of Fortran subroutines for generating box-constrained optimization problems. The technique is a particular instance of that described in Facchinei et al. [1997] and is well suited to large-scale problems. Hopefully, this set of routines will allow researchers to more easily test and compare solution methods for box-constrained optimization with respect to particular problem characteristics. With this in mind, we have developed a flexible code which can be easily linked to most of the existing codes for box-constrained optimization.

Authors' addresses: F. Facchinei, Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, Rome, Italy; J. Júdice and J. Soares, Departamento de Matemática, Universidade de Coimbra, Coimbra, 3000, Portugal.

2. IMPLEMENTATION AND DESIGN

The subroutines have been implemented in ANSI Standard Fortran 77. The user will employ the following routines:

(1) subroutine BCP01: sets the bounds and the data required to evaluate the objective function f and its derivatives.

(2) subroutine BCP02: prints an error message on the screen.

(3) subroutine BCP03: prints information concerning the problem generated.

(4) subroutine BCP10: computes the objective function f at a given point x.

(5) subroutine BCP11: computes the gradient vector of the objective function f at a given point x.

(6) subroutine BCP12: computes the product of the Hessian at a given point x by a vector d.

The box-constrained problem is generated by a call to subroutine BCP01, where the user determines the characteristics of the problem generated through the choice of various parameters. However, the exact definition of the problem also depends on some random choices. For example, the user can determine exactly the number of lower bounds and the number of those that are active at the solution, but their placement is determined randomly. The random-number generator described in Schrage [1979] is included for completeness.

On exit from BCP01, if a flag variable INFRM is not zero then (1) an error has occurred and (2) the box-constrained problem has not been generated. The user may get an appropriate error message by calling subroutine BCP02.

If the problem has been successfully generated (INFRM = 0), the user has access to the vectors L and U containing the lower and upper bounds. Additional information may be obtained through a call to BCP03. Subroutines BCP10, BCP11, and BCP12 provide the value of the generated problem objective function and its derivatives.

The generation of the box-constrained problem is based on an underlying unconstrained problem with objective function g. The data concerning the unconstrained problem must be provided by the user via four Fortran subroutines:

```
SUBROUTINE FUNCTG(N,X,G)
SUBROUTINE GRADG(N,X,GG)
SUBROUTINE HESGD(N,X,D,HGD)
SUBROUTINE SOLUG(N,XBAR)
```

Subroutine FUNCTG gives the value of the function g at a given point x; GRADG gives the gradient of g; and HESSGD performs the product of the

Hessian of g with a given vector. Finally, subroutine SOLUG provides a solution of the unconstrained problem.

A *README* file that accompanies the software provides a more detailed description of all the routines and parameters. The code has been compiled on a SUN SPARC 10, an IBM RS/6000, and an HP Model 712/60. We provide the corresponding *makefiles*.

REFERENCES

FACCHINEI, F., JÚDICE, J., AND SOARES, J. 1997. Generating box-constrained optimization problems. *ACM Trans. Math. Softw. 23*, 3 (Sept.). This issue.

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

Received: September 1995; revised: October 1996; accepted: February 1997