NAG Library Chapter Introduction

e05 – Global Optimization of a Function

Contents

1 Scope of the Chapter

Global optimization involves finding the absolute maximum or minimum value of a function (the *objective function*) of several variables, possibly subject to restrictions (defined by a set of bounds or *constraint functions*) on the values of the variables. Such problems can be much harder to solve than local optimization problems (which are discussed in Chapter e04) because it is difficult to determine whether a potential optimum found is global, and because of the nonlocal methods required to avoid becoming trapped near local optima. Most optimization functions in the NAG C Library are concerned with function **minimization** only, since the problem of maximizing a given objective function F is equivalent to minimizing $-F$. In this chapter, you may specify whether you are solving a minimization or maximization problem; in the latter case, the required transformation of the objective function will be carried out automatically. In what follows we refer exclusively to minimization problems.

This introduction is a brief guide to the subject of global optimization, designed for the casual user. For further details you may find it beneficial to consult a more detailed text, such as Neumaier (2004). Furthermore, much of the material in the e04 Chapter Introduction is relevant in this context also. In particular, it is strongly recommended that you read Section 2.5 in the e04 Chapter Introduction.

2 Background to the Problems

2.1 Problem Formulation

For the purposes of this Library, the global optimization problem is

$$\underset{\mathbf{x}\in R^n}{\text{minimize}}\, F(\mathbf{x}) \qquad \text{subject to} \qquad \mathbf{l}_x \leq \mathbf{x} \leq \mathbf{u}_x \qquad \text{and} \qquad \mathbf{l}_c \leq \mathbf{c}(\mathbf{x}) \leq \mathbf{u}_c, \tag{1}$$

where: $F(\mathbf{x})$ (the *objective function*) is a continuous function; the vectors $\mathbf{l}_x$ and $\mathbf{u}_x$ are elements of $\bar{R}^n$, where $\bar{R}$ denotes the extended reals $R \cup \{-\infty, \infty\}$; and where $\mathbf{c}$ is a vector of m continuous constraint functions $c_1, \ldots, c_m$, with $\mathbf{l}_c$ and $\mathbf{u}_c$ defining the constraints on $\mathbf{c}(\mathbf{x})$. If $m = 0$ the problem is said to be *bound constrained*. Relational operators between vectors are interpreted elementwise. The *feasible region* Φ is the set of all points (*feasible points*) that satisfy all of the constraints. A *solution* of (1) is a feasible point $\hat{\mathbf{x}} \in \Phi$ satisfying

$$F(\hat{\mathbf{x}}) = \min_{\mathbf{x}\in\Phi} F(\mathbf{x}).$$

A *local* minimum minimizes F only on some neighbourhood of $\hat{\mathbf{x}}$. If a local minimum has the smallest objective value over all the local minima, then it is a *global* minimum.

2.2 Terminology

2.2.1 Complete Methods

A global optimization algorithm is called *asymptotically complete* if

(i) assuming indefinitely long run-time and exact computations, a global minimum will be found with certainty (probability one), but

(ii) the algorithm has no way of knowing when a global minimizer has been found.

In comparison, a *complete* method satisfies (i) as well as

(ii(a)) the algorithm is able to recognize a global minimum (to prescribed tolerances) within a finite amount of time.

It is important to appreciate that, for finding a solution exactly, bounds on the amount of work may be very pessimistic. What complete methods guarantee is the absence of any deficiency that would prevent a global minimum from **eventually** being found. To achieve termination with certainty in a finite amount of time, the algorithm requires access to global information about the problem. In the case where only function values are available, as in nag_glopt_bnd_mcs_solve (e05jbc), stopping criteria based on heuristics are present. This is because such a class of method can only terminate with certainty by performing an expensive dense search.

In contrast, *incomplete* methods have intuitive heuristics for searching but no guarantee of not getting stuck near nonglobal, local, minima. Often, to make incomplete methods efficient, expert knowledge on the particular problem class to be solved is required. Examples of incomplete methods are genetic algorithms and simulated annealing.

2.2.2 Branching

Most complete methods recursively split the original problem into smaller, more manageable subproblems. This technique is called *branching*. Branching is usually accompanied by a selection process that splits favourable branches more frequently than others. For example, with *branch and bound* methods, bounds on the objective function for each subproblem are computed in an attempt to eliminate those subregions where no improvement will occur.

Branching methods use a *branching scheme* to generate sequences of sub-boxes that eventually cover the feasible region. At least one function evaluation is made for every sub-box, and new sub-boxes are generated by splitting existing ones. Using appropriate *splitting rules*, convergence to zero of the diameters of sub-boxes is assured. For example, always splitting the oldest box along the oldest side, provided the children do not have too small a volume compared with the parent, guarantees convergence of the method, in the sense described in Neumaier (2004).

Efficiency can be enhanced by carefully balancing global and local searches. While the global part of the search splits sub-boxes with large unexplored territory, the local part usually entails splitting boxes with good function values. For example, the sub-box with the best function value should always be split. A method may also be improved by launching local searches from appropriate candidate local minima.

2.3 Methods of Global Optimization

2.3.1 Multi-level Coordinate Search

The function nag_glopt_bnd_mcs_solve (e05jbc) searches for a global minimizer using branching to recursively split the search space in a nonuniform manner. It divides, or *splits*, the *root box* of the search into smaller sub-boxes. Each sub-box contains a distinguished *basepoint* at which the objective function is sampled. We shall sometimes say ‘the function value of the (sub)box’ as shorthand for ‘the function value of the basepoint of the (sub)box’. The splitting procedure biases the search in favour of those sub-boxes where low function values are expected.

The global part of the algorithm entails splitting sub-boxes that enclose large unexplored territory, while the local part of the algorithm entails splitting sub-boxes that have good function values. A balance between the global and local part is achieved using a *multi-level* approach, where every sub-box is assigned a *level* $s \in \{0, 1, \ldots, s_{\mathrm{max}}\}$. You can control the value of s_{max} using the optional argument **Splits Limit**. Whenever a sub-box of intermediate level $0 < s < s_{\mathrm{max}}$ is split each descendant will be given the level $s + 1$ or $\min\{s + 2, s_{\mathrm{max}}\}$, and the original sub-box’s level is set to 0: a sub-box with level 0 has already been split; a sub-box with level s_{max} will be split no further. This entire process is described in more detail in Section 10.1 in nag_glopt_bnd_mcs_solve (e05jbc), where the *initialization procedure* used to produce an initial set of sub-boxes is outlined, and the method by which the algorithm *sweeps* through levels is discussed. Each sweep starts with the sub-boxes at the lowest level, a process thus forming the global part of the algorithm. At each level the sub-box with the best function value is selected for splitting; this forms the local part of the algorithm.

The process by which sub-boxes are split is explained in Section 10.2 in nag_glopt_bnd_mcs_solve (e05jbc). It is a variant of the standard coordinate search method: the solver splits along a single coordinate at a time, at adaptively chosen points. In most cases one new function evaluation is needed to split a sub-box into two or three children. Each child is given a basepoint chosen to differ from the basepoint of the parent in at most one coordinate, and safeguards are present to ensure a degree of symmetry in the splits.

If you set the optional argument **Local Searches** to be ‘OFF’, then the basepoints and function values of sub-boxes of maximum level s_{max} are put into a ‘shopping basket’ of candidate minima. Turning **Local Searches** ‘ON’ (the default setting) will enable local searches to be started from these basepoints before they go into the shopping basket. The local search will go ahead providing the basepoint is not likely to be in the basin of attraction of a previously-found local minimum. The search itself uses a trust region approach, and is explained in Section 10.3 in nag_glopt_bnd_mcs_solve (e05jbc): local quadratic models are built by a *triple search*, then a linesearch is made along the direction obtained by minimizing the quadratic on a region where it is a good approximation to the objective function. The quadratic need not be positive-definite, so the general nonlinear optimizer nag_opt_sparse_nlp_solve (e04vhc) is used to minimize the model.

3 Recommendations on Choice and Use of Available Functions

The suite of multi-level coordinate search functions (MCS) consist of:

an initialization function:

nag_glopt_bnd_mcs_init (e05jac);

option parameter setting functions:

nag_glopt_bnd_mcs_optset_file (e05jcc),

nag_glopt_bnd_mcs_optset_string (e05jdc),

nag_glopt_bnd_mcs_optset_char (e05jec),

nag_glopt_bnd_mcs_optset_int (e05jfc),

nag_glopt_bnd_mcs_optset_real (e05jgc);

an optional parameter checking function:

nag_glopt_bnd_mcs_option_check (e05jhc);

option parameter getting functions:

nag_glopt_bnd_mcs_optget_int (e05jkc),

nag_glopt_bnd_mcs_optget_real (e05jlc);

and the solver:

nag_glopt_bnd_mcs_solve (e05jbc).

nag_glopt_bnd_mcs_solve (e05jbc) is based on the *multi-level coordinate search* (MCS) method of Huyer and Neumaier (1999). It is an asymptotically complete method for bound constrained problems based on local information (function values) only, employing branching and local searches to accelerate convergence.

Currently there is no function in this chapter for handling constraints that are not bound constraints.

4 Index

supply real optional argument values to nag_glopt_bnd_mcs_solve (e05jbc)
.... nag_glopt_bnd_mcs_optset_real (e05jgc)

5 Functions Withdrawn or Scheduled for Withdrawal

None.

6 References

Huyer W and Neumaier A (1999) Global Optimization by Multi-level Coordinate Search *Journal of Global Optimization* **14** 331–355

Neumaier A (2004) Complete search in constrained global optimization *Acta Numerica* **13** 271–369
