Mathematical Modelling and Analysis 2005. Pages 185–190
Proceedings of the 10th International Conference MMA2005&CMAM2, Trakai
© 2005 Technika ISBN 9986-05-924-0

A PACKAGE FOR DEVELOPMENT OF ALGORITHMS FOR GLOBAL OPTIMIZATION[1]

J. ŽILINSKAS

Institute of Mathematics and Informatics

Akademijos 4, LT-08663 Vilnius, Lithuania

E-mail: `julius.zilinskas@mii.lt`

Abstract. A package for development of algorithms for global optimization is proposed in this paper to ease implementation of covering methods for global optimization. Standard parts of global optimization algorithms are implemented in the package and only method specific rules should be implemented by the user. Some examples of using the proposed package are given.

Key words: Global optimization, branch and bound algorithms

1. Introduction

Global optimization is used to solve practical problems across all branches of engineering, applied sciences and sciences [1]. Application of algorithms to solve practical problems crucially depends on efficiency and reliability of algorithms implementing global optimization methods. However development of such algorithms is not trivial.

The paper presents a C++ package for development of algorithms implementing covering global optimization methods. Global optimization algorithms based on interval arithmetic and balanced random interval arithmetic [8] has been implemented using the proposed package and results are given in the paper.

When computing power of usual computers is not sufficient to solve a practical global optimization problem, the high performance parallel computers may be helpful. Because of that tools for parallelization of global optimization algorithms have been included in the proposed package. A standardized message-passing communication protocol MPI [2] is used for communication between processors. Results of parallelization of interval global optimization algorithm are given in the paper.

[1] The research is supported by NATO Reintegration Grant EAP.RIG.981300.

```
Cover feasible region $D$ by $L = \{L_j | D \subseteq \bigcup L_j, j = \overline{1,m}\}$ using **covering rule**.
$UB(D) = \infty$.
**while** subproblem list is not empty $L \neq \emptyset$ **do**
    Choose $I \in L$ using **selection rule**, exclude $I$ from $L$.
    **if** $LB(I) < UB(D) + \epsilon$ **then**
        Branch $I$ into $p$ subsets $I_j$ using **branching rule**.
        Find $UB(I_j \bigcap D)$ and $LB(I_j)$ using **bounding rules**.
        $UB(D) = \min(UB(D), UB(I_j \bigcap D) | j = \overline{1,p})$.
        $L = \{L, I_j | LB(I_j) < UB(D) + \epsilon, j = \overline{1,p}\}$.
```

Figure 1. General branch and bound algorithm.

2. Covering Global Optimization

Many problems in engineering, physics, economic and other subjects may be formulated as global optimization problems. Mathematically the problem is formulated as

$$f^* = \min_{X \in D} f(X),$$

where $f(X)$ is a nonlinear function of continuous variables $f : \Re^n \to \Re$, $D \subseteq \Re^n$ is a feasible region, n is number of variables. Besides of global minimum f^*, one or all global minimizers $X^* : f(X^*) = f^*$ should be found.

One of the classes of global optimization methodsare covering methods. Covering methods can solve global optimization problems of some classes with guaranteed accuracy. Covering methods detect the sub-regions not containing the global minimum and discard them from further search. The partitioning of the sub-regions stops when global minimizers are bracketed in small sub-regions guaranteeing the prescribed accuracy. A lower bound for the objective function over the sub-region may be used to indicate the sub-regions who can be discarded. Some methods are based on lower bound constructed as convex envelope of an objective function [1]. Lipschitz optimization is based on assumption that the slope of an objective function is bounded [5]. Interval methods estimate the range of an objective function over a sub-region defined by a multidimensional interval using interval arithmetic [4].

A branch and bound technique can be used for managing the list of sub-regions and the process of discarding and partitioning. An iteration of a classical sequential branch and bound algorithm processes a node in the search tree representing a not yet explored sub-region of the feasible region. Iteration has three main components: selection of the node to process, branching of the search tree by dividing the selected sub-region and bounding of the branches by discarding not promising sub-regions. The rules of selection, branching and bounding differ from algorithm to algorithm. The general branch and bound algorithm for global optimization is shown in Figure 1. Before the cycle, the feasible region is covered by one or several partitions whose are added to the list of candidates L.

The rules of covering and branching depend on type of partitions used. The bounding rule describes how the bounds of minimum are found. For the upper bound for minimum over feasible region $UB(D)$ the best currently found value of objective function might be accepted. The lower bound for values of objective function over considered sub-region $LB(I)$ may be estimated using convex envelopes, Lipschitz condition or interval arithmetic.

There are three main strategies of selection:

- Best first – select an element of L with minimal lower bound. Candidate list can be implemented using heap and priority queue.
- Depth first – select the youngest element of L. First-In-Last-Out structure is used for candidate list which can be implemented using stack.
- Breadth first – select the oldest element of L. First-In-First-Out structure is used for candidate list which can be implemented using queue.

Application of algorithms to solve practical problems crucially depends on efficiency and reliability of algorithms implementing global optimization methods. Development of such algorithms is not trivial. However, branch and bound algorithms for global optimization have general scheme and differ only by rules of covering, selection, branching and bounding. Therefore we propose a package for development of algorithms for covering global optimization, where only particular rules should be implemented.

3. Description of the Package for Global Optimization

A C++ package for development of algorithms for covering global optimization methods have been implemented. Only the method specific rules, for example evaluation of bounds for the values of the objective function over the sub-region, should be implemented by the user. The package includes:

- Vector templates to define feasible region and sub-regions.
- Heap and queue templates to define lists of tasks and solutions. Heap is used for list of tasks when ‘best first’ selection rule is used, queue is used for list of solutions and list of tasks when ‘breadth first’ selection rule is used.
- Implementation of timer for measuring speed of algorithms. Time is important criterion to measure performance of global optimization algorithms.
- Implementation of branch and bound algorithm. Parallel Master-Slave version of branch and bound is also implemented.

Sizes of C++ codes implementing the proposed package are given in Table 1.

4. Case Studies

Implementation of the list of candidates is one of important factors of performance of branch and bound algorithms. Implementation of stack and queue for ‘depth first’ and ‘breadth first’ selection is trivial. Time of insertion and deletion of element to/form such type of structure does not depend on number of elements in the list. However the list of candidates for ‘best first’ selection

Table 1. Size of implementations in the package.

Part of package	lines	words	characters
Vector template	119	571	3442
Queue template	139	466	3001
Heap template	161	563	3946
Timer	158	470	3581
Key pressed check	46	114	1005
Branch and bound algorithm	455	1768	13727
Total	1078	3952	28702

requires selection of candidate with the smallest value. Priority queue is often used for this purpose, for example in PROFIL V 2.0 [6] implementation of the global unconstrained minimization method involving a combination of local search, branch and bound technique and interval arithmetic, and in CToolbox (C++ toolbox for verified computing) [3] which is a library for problem-solving routines covering one-dimensional and multi-dimensional problems: accurate evaluation of polynomials, automatic differentiation, linear and non-linear systems of equations, linear optimization, global optimization, and zeros of complex polynomials.

Although time of selection and deletion of element from the priority queue does not depend on the number of elements in the list, the worst case insertion time linearly depends on the number of elements. Because of this priority queue implementation can be usable only for solving small example problems where the largest number of candidates in the list is small. For most of optimization problems the list of candidates grows rapidly and insertion of elements to the list of candidates can take even more time than calculation of bounds which is supposed to be the most time consuming part of branch and bound algorithms. 'Depth first' and 'breadth first' selection strategies can perform better than 'best first' because of efficient implementation of the list of candidates, but not because the number of investigated nodes is smaller. Parallel branch and bound with priority queue implementation of the list of candidates can have better speedup because of reduced time of insertion after distribution of the list of candidates, but not because of excellent load balancing.

Some of the mentioned problems can be at least partly avoided by using heap structure, which is a complete binary tree where each node has larger value of criterion than its parent. Heap and priority queue are implemented in the proposed template. To compare performance of heap and priority queue, lists of different sizes have been constructed inserting elements with random keys. The sum times of construction (insertion of elements) and use (selection/deletion of elements) for different sizes of lists have been measured and are shown in Figure 2. The figure shows that priority queue and 100 times larger heap have similar construction and deletion time. This strongly suggests use of heap structure for implementation of the list of candidates for 'best first' selection.

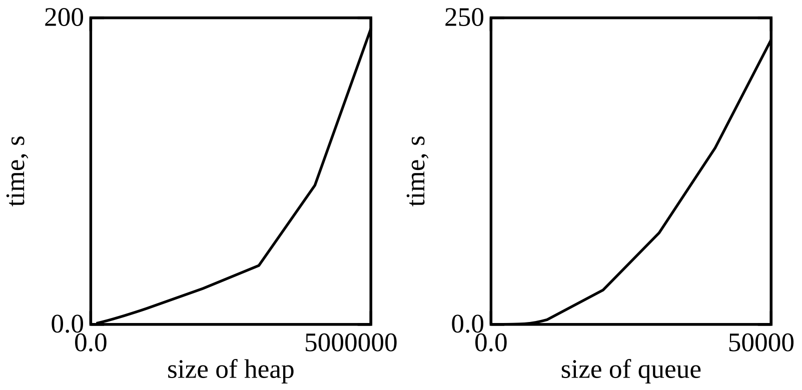

Figure 2. Comparison of performance of heap and priority queue when inserting elements with random keys.

Table 2. Sizes of implementations of algorithms for global optimization.

Algorithm	lines	words	characters
with interval arithmetic	197	755	5917
with balanced random interval arithmetic	202	782	6074
with bounds with controllable tightness	195	776	6781

Global optimization algorithms based on interval arithmetic [4] and balanced random interval arithmetic [8] have been implemented using the proposed package. Algorithms use interval bounds for objective function and interval derivatives for tests of monotonicity and convexity. Sizes of implementations using proposed packages are shown in Table 2. The numbers do not include implementations of interval arithmetic, because it is implemented in independent package. As it can be seen from the numbers, implementation of particular algorithm with the proposed package is 5 times smaller than implementation of the package.

Size of implementation of algorithm with bounds with controllable tightness is also shown in Table 2. Again implementation of this algorithm with the proposed package is 5 times smaller than implementation of the package. This algorithm has been used to investigate influence of tightness of bounds to performance of optimization in [7]. The algorithm uses exact bounds found minimizing and maximizing objective function over considered sub-regions using algorithm with interval arithmetic mentioned above. The investigation is computing intensive, therefore parallelization tools of proposed package has been very helpful. The criteria of parallelization of algorithm for several test functions [7] are shown in Figure 3. For some simple test functions, parallelization efficiency was low. However, these functions can be successfully investigated by a sequential algorithm. For other functions improvement is evident, and parallelization significantly speeds up the experiments.

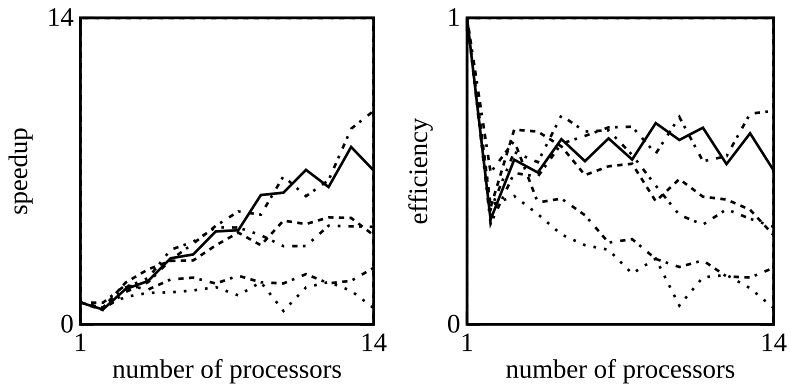

Figure 3. Criteria of parallelization of algorithm with controllable bounds.

5. Conclusions

A C++ package for development of algorithms for covering global optimization methods has been implemented. Heap structure for implementation of the list of candidates for 'best first' selection is preferable because its performance is more than 100 times better than one of priority queue. C++ code implementing particular algorithm with the proposed package is 5 times smaller than C++ code of the package. Parallelization tools of proposed package are very helpful.

References

[1] C.A. Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*, volume 37 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 2000.

[2] Message Passing Interface Forum. MPI: A message-passing interface standard (version 1.1). Technical report, 1995.

[3] R. Hammer, M. Hocks, U. Kulish and D.Ratz. *C++ Toolbox for Verified Computing: Basic Numerical Problems*. Springer, Berlin, 1995.

[4] E. Hansen and G.W. Walster. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 2nd edition, 2003.

[5] R. Horst, P.M. Pardalos and N.V. Thoai. *Introduction to Global Optimization*, volume 48 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 2nd edition, 2001.

[6] O. Knüppel. PROFIL/BIAS V 2.0. Technical Report 99.1, Technische Universität Hamburg-Harburg, 1999.

[7] A. Žilinskas and J. Žilinskas. On efficiency of tightening bounds in interval global optimization. In: *PARA'04 Workshop on State-of-the-Art in Scientific Computing, June 20-23, 2004, Lyngby*, Lecture Notes in Computer Science, 2005.

[8] J. Žilinskas and I. D. L. Bogle. Balanced random interval arithmetic. *Computers & Chemical Engineering*, **28**(5), 839–851, 2004.