A Parallel Software Package for Nonlinear Global Optimization [*]

Chenyi Hu[†], Baker Kearfott[‡], Shanying Xu, and Xiaoguang Yang[§]

Abstract

In this paper, we report a Fortran 90/95 software package, ParaGlobSol, that reliably finds numerical solutions for continuous nonlinear global optimization problems in parallel.

With this package, we have successfully solved some computational intensive real application problems. Superlinear speedup for some application have been observed because of that our parallel implementations can perform much less computations than the sequential implementation for some applications.

Algorithm development, parallel implementations are discussed in this paper. we provide the URL, which has links to the source code, installation and user guide, as well as performance data at the end of this paper.

1 Introduction

ParaGlobSol [3] is a parallel global optimization package we developed recently. It is written in Fortran 90/95 with MPI to perform inter process communication. The package is actually a parallel implementation of the global optimization software GlobSol [2] which based on interval branch-and-bound algorithm. In this introduction, we review interval arithmetic, nonlinear global optimization, interval branch-and-bound algorithm and GlobSol.

1.1 Interval Arithmetic and Notation

Both GlobSol and ParaGlobSol use interval arithmetic [11, 10] that was first introduced by R. E. Moore in the 1960s. To automatically obtain reliable numerical solution bounds, both mathematically and computationally, interval arithmetic uses machine representable intervals as operands in numerical computation.

[*]This research was partially supported by NSF grant CDA-9522157, ARO grant DAAH-0495-1-0250, and Sun Microsystems.

[†]Department of Computer and Mathematical Sciences, University of Houston-Downtown, Houston, TX 77002, USA

[‡]Mathematics Department,University of Louisiana, Lafayette, LA 70504, USA

[§]Institute of Mathematics and System Science, Chinese Academy of Sciences, Beijing, China

Throughout the rest of this paper, we use boldface letters and capital letters to denote interval quantities and vectors, respectively. We use $\underline{x}$ and $\overline{x}$ to denote the greatest lower bound and the least upper bound for an interval variable $\mathbf{x}$, respectively.

With the notation above, interval binary operations can be defined as below:

Definition: Let $\mathbf{a}$, and $\mathbf{b}$ be intervals and op be an elementary operation. Then, $\mathbf{a}$ op $\mathbf{b}$ is the smallest interval which contains a op b for all $a \in \mathbf{a}$ and $b \in \mathbf{b}$.

For example, $[1, 2] + [-1, 0] = [0, 2]$, $[0, 2] - [1, 2] = [-2, 1]$.

For more on interval computation and applications, interested readers may refer [9, 11] and others.

1.2 Nonlinear Global Optimization

A nonlinear global optimization problem over a given domain $\mathbf{X} \subset \Re^n$ can be described as:

$$\text{To find an } X^* \in \mathbf{X} \text{ such that } \phi(X^*) \leq \phi(X) \;\; \forall X \in \mathbf{X},$$

where $\phi(X)$ is a nonlinear mapping from $\Re^n$ to $\Re$. If other conditions such as $C(X) = 0$ and/or $G(X) \leq 0$ should also be satisfied, then we call it constrained optimization.

Numerically solving global optimization problems has fundamental importance in computational sciences and application. Although various results and computer implementations have been published, there are still many challenges in solving general global nonlinear optimization problems. For example, many existing computational algorithms may only find approximations of local optimum rather than global optimum; numerical results provided by floating point arithmetic may not be reliable; and computationally expensive algorithms make it impractical to solve large-scale optimization problems.

1.3 Interval Branch-and-Bound Algorithm

In recent years, algorithms that automatically verify global optimum have been developed [4, 12]. Among them are interval branch-and-bound algorithms. In [7], Kearfott reviewed these algorithms which form the foundation of this paper. With minimal notation and without details of interval arithmetic, we review the basic ideas of interval branch-and-bound algorithms.

Let a interval box $\mathbf{X} = (\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{N-1})$ represent a search region. Then, the principle of branch-and-bound algorithms is to maintain one or more of these lists: $\mathcal{L}$, a list of boxes $\mathbf{X}$ to be processed; a list $\mathcal{U}$ of boxes which have been reduced to small diameter by the algorithm; and a list $\mathcal{C}$ of boxes that have been verified to contain critical points by the algorithm. Boxes in $\mathcal{L}$ can be considered one-by-one for processing. When $\mathcal{L}$ is exhausted, we evaluate ϕ on every box of list $\mathcal{U}$ and on the critical points in list $\mathcal{C}$, and then find the possible global minimizer.

Algorithm 1: (Sequential branch-and-bound)

```
1. Initialize $\mathcal{L}$ by placing the searching domain $\mathbf{X}_0$ in it
2. Initialize the current optimum $\overline{\phi}$ by picking a point $X \in \mathbf{X}_0$, then evaluate $\phi(X)$ with interval arithmetic
3. While $\mathcal{L} \neq \emptyset$ do
    (a) Remove a box from $\mathcal{L}$ for processing
    (b) Compute a lower bound on the range of $\phi$ over $\mathbf{X}$. If $\underline{\phi(\mathbf{X})} > \overline{\phi}$ then discard $\mathbf{X}$ and back to the step 3
    (c) Use interval arithmetic to compute an upper bound on the range of $\phi$ over $\mathbf{X}$, then update $\overline{\phi}$ by $\overline{\phi} \leftarrow \{\overline{\phi}, \overline{\phi(\mathbf{X})}\}$
    (d) Use interval arithmetic to compute $\bigtriangledown\phi(\mathbf{X})$. If $0 \notin \bigtriangledown\phi(\mathbf{X})$ then discard $\mathbf{X}$ and back to the step 3
    (e) For any $X \in \mathbf{X}$, if $\bigtriangledown^2\phi(X)$ cannot be positive definite, then discard $\mathbf{X}$ and back to the step 3
    (f) Apply interval Newton method to possibly do one or more of the following
        • rejecting $\mathbf{X}$
        • reducing the size of $\mathbf{X}$
        • verify existence of a unique critical point in $\mathbf{X}$
    (g) Subdivide $\mathbf{X}$ if step (f) the above did not result in a sufficient change in $\mathbf{X}$, then return all resulting boxes to $\mathcal{L}$ for subsequent processing.
```

1.4 GlobSol

Kearfott published GlobSol, a Fortran 90 package, that implemented interval branch-and-bound method. By applying powerful software tools such as automatic differentiation and LINPACK, GlobSol provides reliable bounds of global nonlinear optimization problems with/without constraints. Many real application problems have been successfully solved with GlobSol. For more information, please see GlobSol [2] homepage.

2 Parallel Algorithm

2.1 Motivation

With GlobSol, many application problems have been solved successfully. However, computation time required in solving some applications are enormous. For example, GlobSol used 2342 CPU seconds on a Sun Ultra workstation to solve the four dimensional constrained nonlinear optimization problem for a T-bill trading application of Bank One. A Sun Ultra spent 4320 CPU seconds and processed 5,000 boxes still could not find the optimum for a Swiss Bank currency trading problem.

To improve the overall performance of GlobSol, we have developed its parallel implementation on distributed multicomputers.

2.2 Parallel Interval Branch-and-Bound Algorithm

The main computation of interval branch-and-bound algorithm is the step 3 of Algorithm 1. We noticed that the processing of each existing box in the list $\mathcal{L}$ is computational independent. Therefore, we may use multiple processors to process boxes in $\mathcal{L}$ concurrently. Here is our basic parallel algorithm.

Algorithm 2: (Parallel branch-and-bound)

1.
   - *One process* initializes the list $\mathcal{L}$ by placing the searching domain $\mathbf{X}_0$ in it, and initialize the current optimum $\overline{\phi}$ by picking a point $X \in \mathbf{X}_0$, then evaluate $\phi(X)$ with interval arithmetic.
   - *Other processes* idle.
2.
   - *The process* which performed initialization distributes boxes in its list $\mathcal{L}$, and $\overline{\phi}$ to other processes
   - Other processes receive boxes and $\overline{\phi}$ from process 0 and form their local $\mathcal{L}$ list.
3. *All processors* do while local $\mathcal{L} \neq \emptyset$
   (a) Remove a box $\mathbf{X}$ from $\mathcal{L}$ for processing
   (b) Compute a lower bound on the range of ϕ over $\mathbf{X}$. If $\underline{\phi(\mathbf{X})} > \overline{\phi}$ then discard $\mathbf{X}$ and back to the step 3
   (c) Use interval arithmetic to compute an upper bound on the range of ϕ over $\mathbf{X}$, then update $\overline{\phi}$ by $\overline{\phi} \leftarrow \{\overline{\phi}, \overline{\phi(\mathbf{X})}\}$
   (d) Use interval arithmetic to compute $\bigtriangledown\phi(\mathbf{X})$. If $0 \notin \bigtriangledown\phi(\mathbf{X})$ then discard $\mathbf{X}$ and back to the step 3
   (e) For any $X \in \mathbf{X}$, if $\bigtriangledown^2\phi(X)$ cannot be positive definite, then discard $\mathbf{X}$ and back to the step 3
   (f) Apply interval Newton method to possibly do one or more of the following
      - rejecting $\mathbf{X}$
      - reducing the size of $\mathbf{X}$
      - verify existence of a unique critical point in $\mathbf{X}$
   (g) Subdivide $\mathbf{X}$ if step (f) the above did not result in a sufficient change in $\mathbf{X}$, then return all resulting boxes to $\mathcal{L}$ for subsequent processing.

4.
   - *One process* receives other processes local optimum and finds the global optimum.
   - *All other processes* send their local optimum to the process above.

Algorithm 2 is a coarse-grained, SPMD data-parallel algorithm. All processes actually run GlobSol with their local data.

3 Implementations

We have made two implementations, ParaGlobSol 0.1 and 0.2 for the above parallel algorithm. We have successfully run both implementations on networked SUN Ultra stations, and solved all test problems that can be solved by the sequential GlobSol.

To make our implementations more efficient, we tried different schemes to balance workload and take the advantage of parallel computing to eliminate computations. ParaGlobSol 0.1 balances workload through equally distributing initial boxes to all processes, while ParaGlobSol 0.2 dynamically balances workload by distributing one box at a time to the process which requires a box to process.

3.1 Load Balancing

To achieve high efficiency in our implementation, we balance workload by distributing the boxes in $\mathcal{L}$ in different schemes. The simplest scheme probably is to count the number of boxes in the $\mathcal{L}$ list, and then divide it by the number of processes (say the ratio is m). In the step 2 of Algorithm 2, we may distribute m boxes to other processes. This scheme is applied in ParaGlobSol 0.1.

Although every process has almost equal number of boxes, it usually requires different amount of time to complete its computation, since some boxes maybe rejected and new boxes can be created. Therefore, we developed a dynamic scheme which better balances the workload. We implemented the dynamic load balancing scheme below in ParaGlobSol 0.2.

Dynamic Workload Balancing:

1. The initialization process distributes only one box to other processes
2. Every process works with the box it receives.
3. Whenever a process completes its work, it requests a new box, if exists, from other processes.

3.2 Computation Reduction

In our parallel implementation, we have tried to reduce the amount of computations through parallelly updating local lists frequently to purge boxes that are no longer needed for further processing. We describe this idea in detail below.

In the step 3(b) of Algorithm 1, the sequential branch-and-bound method, a box will be discarded if the lower bound of the function value over the box is greater than current optimum. The current optimum is updated sequentially as every box pop up from the list $\mathcal{L}$.

In parallel processing of Algorithm 2, each process has its local current optimum. By frequent all-to-all communication, we can let every process to have current global optimum. Every process works with current global optimum instead of its local optimum. This may eliminate boxes in early stages and significantly reducing total amount of computation. This also explains the superlinear speedup we observed for some applications.

4 Applications

We report two applications of ParaGlobSol that achieved superlinear speedup.

4.1 Portfolio management

For portfolio managers, the basic problem they face is to make the combination of securities under various investment constraints (cash flow, risk elements, maturity structure, corporate investment policies, etc.) in order to maximize the return. These parameters are usually within ranges rather than points. Therefore, it is appropriate to use interval methods.

In [1], G. Corliss and B. Kearfott reported that P. Thalacker and *et al* found the best asset allocation for two sample fixed-income portfolios of BancOne with GlobSol. All of the constraints were satisfied.

One of the fixed income portfolios contains only 3-, 6-, and 12-month Treasury bills. Actual spot price data for each type of T-bill from U. S. Treasury auction were used. Historical spot price data for each bill were also used to calculate the variances of each type of T-bill and covariances between bills. GlobSol calculated the optimal asset allocation to be about: 10% in 3-month T-bills, 60% in 6-month T-bills, and 30% in 12-month T-bills. The rate of return is 5.363% per year.

Another portfolio contains only 1-, 5-, 10-, and 30-year Treasury bonds with four investment constraints: average duration, cash flow, variance, and total value. Historical data were used to calculate T-bond variances and covariances between bonds. Hypothetical values were chosen for spot prices, coupon payments, and face values. For a typical set of constraints, GlobSol determined the optimal asset allocation to be about: 4% in one year bounds, 65% in 5-year bounds, 4% in 10-year bounds, and 26% in 30-year bounds. Optimal rate of return is about 7.388%.

GlobSol solved the problem in 2352.7 CPU seconds on a Sun Ultra-1 station. With ParaGlobSol [3], the problem was solved in 15.75 CPU seconds with 9 processes. The extraordinary speedup was due to significant reduction of total amount of computation in parallel execution.

4.2 Risk control in currency trading

The objective of the portfolio risk control system for Swiss Bank currency trading is to determine the value at risk (VaR), or the maximum overnight loss the bank may face on the portfolio of trades, assuming that the portfolio is illiquid overnight. The maximum loss is determined for the portfolio closing position by allowing the risk factors to change and calculating the price sensitivities of the trades to these changes in the risk factors. Total value of the portfolio is evaluated in the light of several risk factors. Since these factors are allowed to change within specified ranges, finding global maximum becomes quite difficult without using interval methods.

S. Ranjit and *et al* applied GlobSol and found the minimum variance to determine the VaR for portfolios containing calls and puts on 2, 4, and 7 exchange rates [1]. The portfolio contains Swiss Franks, Mexican Pesos, S. Korea WONs, Australian Dollars, Netherlands Guilder, and Singapore Dollars. Garman-Kohlhagen model for valuing foreign exchange option, were used. Data used were one two week period in October, 1997. GlobSol found the minimum variance portfolio should be 74.7% Mexican Peso, 8.9% Australian Dollar, 8.4% Netherlands Guilder, and 8.0% Singapore Dollars. The minimum variance for that time period is 0.00055% of the portfolio value, while the variances of the currencies range from 0.0013% to 0.049%.

GlobSol solves the problem in over 4320 CPU seconds on a Sun Ultra-1 station. With ParaGlobSol [3], the problem was solved in 49.95 CPU seconds with 9 processes. The superlinear speedup was due to significant reduction of total amount of computations in parallel execution.

In the table below, we summerize the performance data for these applications.

Risk control			Portfolio management		
# of process	CPU Time	Speedup	# of process	CPU Time	Speedup
1	4320	N/A	1	2352.7	N/A
7	54.34	79			
9	49.95	86	9	15.75	149

As we explained in the previous sections, the superlinear speedup were reached since our parallel implementation significantly reduced the total computation in these applications.

5 Conclusion

Interval branch-and-bound algorithms have their unique feature for nonlinear global optimization such as being able to take interval parameters and to verify global optimum automatically. However, in practice, we need to apply parallel computing technology to improve the overall efficiency.

ParaGlobSol is a prototype parallel implementation of interval branch-and-bound algorithm with Fortran 95 and MPI. It has achieved superlinear speedup for some application problems. Source code of ParaGlobSol, installation and user guide, as well as sample applications are freely available at the URL:

http://www.mscs.mu.edu/∼globsol/ParaGlobSol.html

References

[1] G. Corliss, and B. Kearfott, *Rigorous Global Search: Industrial Applications*, J. Reliable Computing, pp 1-16, 1999.

[2] *GlobSol*, `http://www.mscs.mu.edu/∼globsol/`

[3] C. Hu, *ParaGlobSol*, `http://www.mscs.mu.edu/∼globsol/ParaGlobSol.html`

[4] E. R. Hansen, *Global optimization using interval analysis*, Marcel Dekker, Inc., New York, 1992.

[5] C. Hu, A. Frolov, B. Kearfott, and Q. Yang, *A general iterative sparse linear solver and its parallelization for interval Newton's methods*, J. Reliable Computing, No. 1(3), pp. 251-264, 1995.

[6] C. Hu, *Parallel Solutions for Large-Scale General Sparse Nonlinear Systems of Equations*, J. Computer Science and Technology, Vol. 11, No.3, pp. 257-271, 1996.

[7] R. B. Kearfott, *A review of techniques in the verified solution of constrained global optimization problems*, in Applications of Interval Computations, Kluwer Academic Publishers, 1996.

[8] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, 1996

[9] R. B. Kearfott, M. Dawande, K. Du, C. Hu, *Algorithm 737: INTLIB: A portable Fortran-77 interval standard-function library*, *ACM Trans. Math. Software*, Vol. 20, No. 4, pp. 447-459, 1994.

[10] R. E. Moore, *Interval arithmetic and automatic error analysis in digital computing*, Ph. D. thesis, Stanford University, 1962.

[11] R. E. Moore, *Methods and applications of interval analysis*, SIAM, 1979.

[12] H. Ratschek and J. Rokne, *New computer methods for global optimization*, Wiley, New York, 1988.