Set Inversion Via Interval Analysis: A Study on Parallel Processing Implementation[*][†]

K. A. Nasiotis[1,2], D. López[2], S. P. Adam[1], and L. G. Casado[2]

[1]Dept. of Informatics and Telecommunications, University of Ioannina, Greece
[2]Supercomputing Group, University of Almería (CeiA3), Spain

Keywords: Interval analysis; Set membership techniques; Parallel branch-and-bound

Introduction

Among the success stories of interval computation [5] one may report the application of interval based global optimization, validation of numerical calculations, the use of intervals for modeling uncertainty and dealing with uncertain systems, etc.

This work focuses on Set Inversion Via Interval Analysis (SIVIA) [4] a method used for solving problems such as nonlinear parameter, state or error estimation for systems operating under bounded uncertainty. The method is primarily a set membership technique designed to solve such estimation problems provided that system operation is described by some analytic function $f : X \subseteq \mathbb{R}^n \to Y \subseteq \mathbb{R}^m$ for which some suitable interval extension $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ can be defined. Then, given an interval vector, i.e. a box, $[y] \subseteq Y$, one needs to determine the set of unknown vectors $x \in X$ such that $f(x) \in [y]$. SIVIA starts with an initial box $[X_0]$ such that $X \subseteq [X_0] \subseteq \mathbb{IR}^n$ and computes an approximation of the set of interest $S = \{x \in X \subseteq \mathbb{R}^n | f(x) \in [y]\} = f^{-1}([y]) \cap X$ as a union of axes aligned boxes.

Computation explores the search space $[X_0]$ applying a branch-and-bound, or more precisely, given the type of processing, a branch-and-prune (B&P) strategy whose performance depends on the size of the problem i.e. the size of the search space $[X_0]$, its dimension, the function f itself, the distribution of the vectors of interest $x \in [X_0]$ and the "resolution" adopted for the approximation of the set S. SIVIA has been successfully applied in control systems problems with few parameters. For problems with higher dimensions, larger sized input space and fine "resolution" the performance of SIVIA deteriorates severely and becomes practically inapplicable. Hence, the need to investigate the possibility of a parallel implementation towards obtaining some affordable computational cost.

Background and Related Work

Interval computations have proven to be extremely important to a number of problems for which errors produced by calculations or due to uncertainty that can be modeled in terms of intervals. However, Kreinovich [6] provides a number of convincing arguments that interval computations are NP-hard and the only way to deal with NP-hardness is to use parallel versions of the algorithms, when this is feasible.

SIVIA itself relies on interval computations and it is also known to suffer from the curse of dimensionality which is inherent to its B&P processing style. A recent example of this argument is reported by the work of Adam et al. [1] who formulated the problem of estimating generalization of a multilayer perceptron as a parameter estimation problem and used SIVIA for exploring search spaces such as $[-1, 1]^{10}$. The sequential implementation of SIVIA in such experiments gave extremely interesting results but at the same time it proved the practical impossibility of SIVIA

[*]This paper has been supported by the Spanish Ministry (RTI2018-095993-B-100), in part financed by the European Regional Development Fund (ERDF).

[†]Konstantinos A. Nasiotis placement in (CeiA3) is financed by the ERASMUS+ Program.

to cope with higher dimensions. This deficiency led to the definition of the so-called contractor programming in order to diminish the size of the boxes explored by SIVIA [3], while some practitioners tried to effectively parallelize SIVIA [7, 8].

As reported in [8], Marvel et al. used SIVIA for estimating model parameters in biological systems, under bounded conditions of uncertainty. They adopted a parallel implementation using multiple processing cores and they developed a method for use on a single multicore workstation using POSIX threads to process subsets of the parameter space while access to shared information was controlled by mutex-locked linked lists. The results obtained for two biological models, namely, the nonlinear Lotka-Volterra predator-prey model and the SEIR infectious disease model, using 8 threads on an 8-core machine, seem to be satisfactory but it remains unclear what the speedup will be when scaling up the implementation to 16, 32 or more more CPUs.

Another work that merits to be cited here is the effort of Le Ménec [7] for computing, in real time, the viability kernel as a tool for decision making in autonomous systems. To this end he applied an interval based algorithm based on SIVIA for computing the viability kernel of the underlying nonlinear system. The author proposes a parallel implementation of SIVIA on a multi-core processing system while making use of the contractors concept. The parallel implementation scheme adopted in this work, [7], uses a pool of jobs each one corresponding to the evaluation of a hyper-box. A host streams the jobs to a multicore system assigning a job to each available core while recovering the results of each completed job. This implementation is a possible way to deal with the inherent deficiency of the IBEX library to share objects between threads i.e. being thread unsafe. Moreover, the author does not present any performance results and so it is not possible to objectively criticize this work.

In the work presented, herein, we adopt a more elaborated parallelization strategy which uses an Asyncronous Multiple Pool of boxes and a dynamic load balancing scheme based on workload estimation for optimizing the use of the computing resources.

Proposed Parallel Version

Parallelization of B&B style algorithms were widely studied in the literature, see [2] and references therein. In order to simplify the algorithm let us focus on SIVIA when m=1. So, we are dealing with determining the "level set" of the function f in the (one dimensional) interval $[y]$. Then, for $f(x) = x_1^2 + x_2^2$, $[y] = [1, 2]$, $[X_0] = [-1.5, 1.5]$ and $\epsilon = 0.05$ the result of the inversion operation is given in Figure 1. In this specific implementation

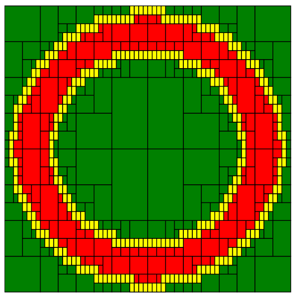

Figure 1: Circle level set in [1,2] interval. Red: in , yellow: border, green: out.

of SIVIA the selected box is divided by the widest dimension and Depth-First is used to select next box.

Our parallel implementation is based on a Asynchronous Multiple Pool of boxes using a dynamic number of threads [2] in C-XSC which was suitably modified to be thread safe. Dynamic load balancing is provided by generating a new thread, if the current number of threads is smaller than a threshold, and by moving the next selected box (the smallest one) to the new thread. Initial experiments run on a node of Bullion S8 with 8 Intel(R) Xeon(TM) E7 8860v3 @ 2.20GHz (16 cores) and 2.3TB of total RAM. The results, in number of boxes, for the circle problem, shown in Figure 1 for $\epsilon = 10^{-6}$ are: $|L_{in}|$=13,501,140, $|L_{border}|$=40,503,776 and $|L_{out}|$= 13,501,276 and the total number of evaluated boxes is 94,508,607. L_{in}, L_{out} and L_{border} are the lists of the boxes that are inside, outside or on the ϵ-border of the set discovered by SIVIA.

We also tested the parallel algorithm for the Griewank function using the same hardware, $[y]$=$[1.5, 3]$, $[X_0]$=$[-10, 10]^2$ and ϵ=10^{-6}.

$$\sum_{i=1}^{2} \frac{x_i^2}{4000} - \prod_{i=1}^{2} cos(\frac{x_i}{\sqrt{i}}) + 1$$

The results obtained are: $|L_{in}|$=122,424,640, $|L_{border}|$=343,639,512 and $|L_{out}|$=122,445,084 and the total number of evaluated boxes is 833,378,959. The speedup results for the above experiments are displayed in Figure 2, hereafter.

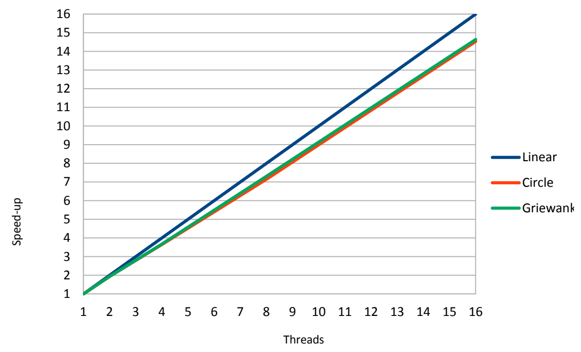

Figure 2: Speedup achieved by the current implementation for the two examples

The number of evaluated boxes increases not only with n but with the type of the problem, as well. The achieved, close to linear, speedup is due to a the dynamic load balancing used, which is based on the estimation of the workload of the boxes pending for evaluation. This allows to select half of the pending workload in order to migrate those boxes to a new thread. Additionally, a Cut-off of the parallelism is used in order to avoid migration of boxes without significant workload (subtree). We expect this strategy to further improve both the parallel version and the sequential one, as the number of evaluated boxes will be reduced.

Conclusion

Despite its efficiency for small-sized estimation problems SIVIA is NP-hard and fails to address problems of larger size in reasonable time. The aim of this study is to investigate an efficient parallel implementation of SIVIA on multi-core systems. The results obtained, so far, support the prospect of acceptable processing times while underlining the need of focusing on techniques accelerating the parallel version shown here. Scaling up the current implementation to systems with a larger number of cores is of primary concern for this study.

References

[1] Stavros P. Adam, Aristidis C. Likas, and Michael N. Vrahatis. Evaluating generalization through interval-based neural network inversion. *Neural Computing and Applications*, Mar 2019.

[2] Juan F. R. Herrera, José M. G. Salmerón, Eligius M. T. Hendrix, Rafael Asenjo, and Leocadio G. Casado. On parallel branch and bound frameworks for global optimization. *Journal of Global Optimization*, 69(3):547–560, Nov 2017.

[3] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. Interval analysis. In *Applied Interval Analysis*, pages 11–43. Springer, 2001.

[4] Luc Jaulin and Eric Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053 – 1064, 1993.

[5] Ralph B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1):95–112, 1996.

[6] Vladik Kreinovich and Andrew Bernat. Parallel algorithms for interval computations: An introduction. *Interval Computations*, 1994, 01 1994.

[7] Stéphane Le Ménec. Interval Based Parallel Computing of the Viability Kernel. https://swim2016.sciencesconf.org/data/pages/LeMenec.pdf, 2013. SWIM 2016 – Lyon.

[8] Skylar W. Marvel, Maria A. de Luis Balaguer, and Cranos M. Williams. Parameter Estimation in Biological Systems Using Interval Methods with Parallel Processing. In *Proc. of the 8th International Workshop on Computational Systems Biology*, pages 161–168, Zürich, Switzerland, 2011.