

# NEW LOAD BALANCING CRITERION FOR PARALLEL INTERVAL GLOBAL OPTIMIZATION ALGORITHMS. \*

LEOCADIO G. CASADO and INMACULADA GARCIA

Department of Computer Architecture and Electronics,  
Almería University, 04120, Almería, Spain.

## ABSTRACT

In this paper we present evaluations of mapping and load balancing methods for a type of irregular data parallel applications on distributed memory multiprocessors. Problems we deal with are nonuniform and dynamic and they come from the field of global optimization. They belong to the class of interval branch and bound methods, so their parallel implementations are not straight forward and seem to require the use of dynamic load balance methods. Comparison between a static and a dynamic mapping strategy for this general problems are carried out.

**keywords:** Parallel and Distributed Processing, Load Balancing, Global Optimization, Branch-and-Bound.

## 1 INTRODUCTION

A wide variety of problems coming from several different scientific and engineering fields consists in determining the optimum values of a set of parameters which are responsible for the stability or effectiveness of a system. This kind of problems can be formulated as a Global Optimization Problem (GOP); i.e. as that of minimizing a continuous function  $f$ . The problem is stated as follows. Given a continuous and non-linear function  $f : S \subset R^n \rightarrow R$ , to find the set of minimizer points  $x^* \in S$  such that  $f(x^*) \leq f(x) \forall x \in S$ .

During the last few years several stochastic and deterministic GOP methods has been proposed [10, 8, 9]. This paper will only deal with a deterministic method which belongs to the well known Branch and Bound (B&B) methods. One of the main advantages of using Branch and Bound methods consists in their reliability in obtaining a full knowledge of the location of all the global minima of a function  $f$ . They are frequently used as an intelligent search method in the context of Global Optimization (GO). The main

drawback of B&B methods is their higher computational cost compared to stochastic methods. Nevertheless B&B methods exhibit a high degree of parallelism which can be exploited for improving the efficiency of the search procedure which inherently appears in the GO problems. The computational power needed to solve GO problems using B&B methods and their high degree of potential parallelism make them suitable candidates to be solved in a multiprocessing environment. Nevertheless Branch and Bound algorithms are irregular and have an almost unpredictable behavior and data dependencies; i.e. the type of problems we are going to deal with are irregular and dynamic. A wide set of parallel B&B models for shared and distributed memory multiprocessing systems have been proposed [1, 2, 3, 6, 7]. Herein, only the distributed solution is considered.

This work investigates the improvements obtained by a new parameter, called  $pf^*$ , which gives a valuable estimation of how close or far is an interval from the region of attraction to a local or global minimizer point [4, 5]. Taking into account that in B&B most of the computational work is made around the region of attraction to a minimizer point, this parameter  $pf^*$  can be used as a predictor of the computational cost of a particular subproblem (very useful for implementing a good load balance) as well as an estimator of the probability of a subinterval for containing a minimizer point of the function  $f$ .

This work is organized as follows. In section 2 a short description of the Interval Branch and Bound algorithm and a general computational model for this kind of problems is made. This section is also devoted to describe the three parallel models (static, semi-static and dynamic) which have been implemented and evaluated. Section 3 shows results of the performance evaluations of these proposals.

---

\*This work was supported by the Ministry of Education of Spain (CICYT TIC96-1125-C03-03) and by the Consejería de Educación de la Junta de Andalucía (07/FSC/MDM).

## 2 B&B ALGORITHM AND PARALLEL MODELS

In this work a modified version [4, 5] of the Moore-Skelboe B&B algorithm [12] without additional accelerating devices, (with the exception of the cut-off test) is used. This algorithm uses Interval Arithmetic as a tool for defining an inclusion function  $F$  which encloses the range of  $f$  over an interval  $X$  (also called box); i.e.  $f(x) \in [\underline{F}(X), \overline{F}(X)]$ ;  $\forall x \in X$ . This B&B algorithm can be characterized by the following five rules: **Termination rule**: The algorithm ends when the width ( $w(X) = \overline{X} - \underline{X}$ ) of all the boxes is smaller than an established accuracy parameter  $\epsilon$ . The **Branching rule** consists in a uniform bisection in all the  $n$  dimensions of the problem. **Bounding rule** is given by the lower bound of the interval range of a box ( $\underline{F}(X)$ ). The **Elimination rule** determines that an interval does not contain a feasible solution if  $\underline{F}(X) > \tilde{f}$ , where  $\tilde{f}$  is the current smaller value of  $f$  ( $f$  is evaluated at the center of the subintervals). Two models for the **Selection rule** have been applied: A standard Best-First search, based on the value of the lower bound  $\underline{F}(X)$ , and a hybrid model which uses a Best-First search based on the value of  $pf^*(X) \times w(X)$ , as an estimator of the work load of the box  $X$ , followed by a Depth-First search based on the value of  $pf^*(X) = (\tilde{f} - \underline{F}(X)) / (\overline{F}(X) - \underline{F}(X)) \in [0, 1]$ . This selection rule can be seen as a guided local search that allows to obtain a fast estimation of an optimal solution ( $\tilde{f} \simeq f^* = \min f(x) : x \in S$ ).

From a computational point of view this algorithm can be modeled as a  $2^n$ -ary tree, where nodes and leaves represent all the subintervals built and evaluated from  $S$  (root node). At any stage of the algorithm, a leaf is a candidate either to be rejected or to be subdivided in  $2^n$  subintervals. So, the computational cost of the algorithm and the shape of the final tree strongly depend on the specific function  $f$  (data dependent).

In this work three parallel models (static, semi-static and dynamic) of the B&B algorithm on a parallel computer with  $NP$  processors (PEs) are presented. These parallel models use a direct initialization, where each PE directly computes its own local root nodes. These nodes are taken from a certain depth of the search tree, such that the number of nodes at this depth is higher than the number of PEs.

In the static model a blind and decentralized mapping model is used. Each processor applies a B&B algorithm on its own initial subintervals, using a standard Best-First selection rule, and only information about the optimal solution is exchanged between processors. This static model has been implemented in order to highlight the level of imbalance of the prob-

lems used in this work.

The semi-static mapping uses a centralized model and consists in two stages. The first stage is intended to find a good estimation of the optimal solution. It starts as the static one and finishes after the above mentioned hybrid selection rule (local search) is applied over the initial boxes. Only when a local search finishes the improved  $\tilde{f}$  is broadcasted and the received messages are used to update the state of the local tree by the elimination rule. The remaining boxes in each PE are sent to the central processor. The central processor makes a estimation of the total work load as  $WL_t = \sum_i pf^*(X_i) \times w(X_i)$ ,  $\forall X_i$  in the work tree and it distributes a set of boxes with a  $WL_t/NP$  load among the PEs. At the second stage all the processors apply a B&B algorithm using the hybrid selection rule on the set of subintervals received from the central processor. The algorithm ends when all the processors send to the central one their final tree. In this model only once redistribution of the workload is made.

The dynamic model starts with a direct initialization. As in the semi-static model, it always applies a hybrid selection rule. If a PE run out of boxes, it asks to the central processor for boxes. If the central processor has no boxes, it asks to the rest of PEs for boxes. A PE will return the second, fourth, ... boxes saved in its own priority tree (ordered by  $pf^*(X) \times w(X)$ ) to the central processor. When the central processor receives boxes it sends back to the waiting PEs a load approximately equal to  $WL_t/(NP - 1)$ . This value remains constant until the central processor asks for more boxes. The program finishes when both the central processor and all PEs have no boxes in their work tree.

## 3 RESULTS AND CONCLUSIONS

The static, semi-static and dynamic models for the B&B algorithm have been implemented using PVM and run on a PC multiprocessor based system with up to 8 processing elements. Performance evaluations for the parallel B&B algorithm are shown in table 1. A set of four test functions, Goldstein-Price (GP), Hartman-3 (H3), Levy-3 (L3) and Six-Hump-Camel-Back (C), standard in the literature of GO, have been used in this work. Numerical results of the CPU time, speed-up, number of interval and real evaluations, as well as imbalance (IBL) measurements are shown. All values shown in table 1 are the average of numerical results in five executions. The workload imbalance is defined as  $IBL = (I_{max} - I_{av})/I_{av} \in [0, NP - 1]$ , where  $I_i$  is the number of interval function evaluation done by the processor element  $PE_i$ ,  $I_{max} = \max\{I_i\}$  and  $I_{av} = \sum_i I_i/NP$ ,  $1 \leq i \leq NP$ .

Static					Semi-Static				Dynamic			
NP	1	3	5	7	1	3	5	7	1	3	5	7
CPU Time (sec.)												
GP	130.20	70.80	90.39	38.26	106.52	38.64	23.17	16.60	105.45	35.78	21.76	15.07
H3	961.21	955.42	955.34	948.34	877.77	330.50	234.74	235.31	882.94	296.60	177.58	126.60
L3	401.06	223.19	129.85	90.91	398.91	139.00	114.23	71.70	398.96	133.62	80.28	57.40
C	20.53	9.77	9.76	9.70	16.83	8.27	4.98	3.6	16.92	5.83	3.61	2.51
Speed-Up												
GP	1.00	1.84	1.44	3.40	1.00	2.75	4.60	6.42	1.00	2.95	4.85	6.99
H3	1.00	1.01	1.01	1.01	1.00	2.66	3.74	3.73	1.00	2.98	4.97	6.97
L3	1.00	1.80	3.01	4.41	1.00	2.87	3.49	5.56	1.00	2.99	4.97	6.95
C	1.00	2.10	2.10	2.12	1.00	2.04	3.38	4.68	1.00	2.90	4.69	6.74
IBL												
GP	0.00	0.61	2.47	1.10	0.00	0.05-0.08	0.01-0.05	0.01-0.04	0.00	0.01	0.02	0.02
H3	0.00	1.98	3.96	5.94	0.00	0.04-0.13	0.06-0.34	0.15-0.88	0.00	0.01	0.01	0.01
L3	0.00	0.65	0.62	0.55	0.00	0.18-0.04	0.18-0.38	0.11-0.19	0.00	0.03	0.03	0.03
C	0.00	0.42	1.34	2.15	0.00	0.10-0.45	0.04-0.40	0.03-0.40	0.00	0.01	0.03	0.04
Interval Function Evaluations												
GP	169209	169310	169264	169293	169213	169354	172899	172902	169213	169378	169363	165959
H3	283753	284117	284322	284382	283753	284024	284073	284144	283753	283954	283931	284162
L3	32765	32764	32760	32760	32765	32808	35443	35308	32765	32844	32964	33030
C	33413	33420	33674	34774	33473	34048	37226	37364	33449	33510	33692	32416
Real Function Evaluations												
GP	42302	42326	42312	42319	42603	42322	42200	42201	42303	42343	42337	41485
H3	35469	35514	35539	35546	35469	35439	35445	35454	35469	35493	35490	35519
L3	8191	8190	8186	8186	8191	8186	7836	7803	8191	8210	8237	8253
C	8353	8354	8414	8689	8368	8496	8282	8317	8362	8376	8419	8100

Table 1: Numerical results. IBL for the semi-static version are those obtained at the first and second stage of the algorithm.

In these implementations we have been working with two main drawbacks: a set of computationally un-expensive functions (the set of test functions has been implemented in such way that their computational cost is higher than  $10^{-3}$  sec., but smaller than  $10^{-2}$  sec.) and the slowness of the communications of the distributed system (a network of PCs).

Our results show that the number of real and interval function evaluations for parallel versions of the algorithm does not increase significantly compared to the sequential version. Only in the semi-static version, a slight increasing of the number of function evaluations has been reported. A global evaluation of the results for speed-up and IBL for the three parallel versions show that the semi-static version improves the workload balance as well as the speed-up, but the dynamic version obtains better results, with values of IBL 10 times smaller than the semi-static ones.

## References

- [1] S. Berner, Ein paralleles Verfahren zur verifizierten globalen Optimierung (in German), *PhD Thesis*, 1995, University of Wuppertal, Germany.
- [2] O. Caprani, B. Godthaad and K. Madsen, Use of a Real-Valued Local Minimum in Parallel Interval Global Optimization, *Interval Computation*, 2, 1993, 71–82.
- [3] J. Erikson and P. Lindström, A Parallel Interval Method Implementation for Global Optimization using Dynamic Load Balancing, *Reliable Computing*, 1, 1995, 77–91.
- [4] L. G. Casado, I. García and T. Csendes, Adaptive Multisection in Interval Methods for Global Optimization, 1996.
- [5] L. G. Casado, I. García and T. Csendes, New Heuristic Rejection Criterion in Interval Global Optimization Algorithms, 1997.
- [6] T. Henriksen and K. Madsen, Parallel Algorithms for Global Optimization, *Interval Computation*, 3(5), 1992, 87–95.
- [7] A. Leclerc, Parallel interval global optimization and its implementation in C++, *Interval Computations*, 3, 1993, 148–163.
- [8] R. Horst and P. M. Pardalos, (eds.), *Handbook of Global Optimization*. Kluwer, Dordrecht, 1995.
- [9] A. Törn and A. Žilinskas, *Global Optimization*. Springer-Verlag, Berlin, 1987.
- [10] R. Horst and H. Tuy, *Global Optimization. Deterministic Approaches*. Springer-Verlag, Berlin, 1996.
- [11] O. Knüppel, BIAS — basic interval arithmetic subroutines, *Technical Report 93.3*, University of Hamburg, 1993.
- [12] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester, 1988.