CS-6235
(Project Proposal)

Global Extrema locator using interval arithmetic

Advisor : Professor Ganesh Gopalakrishnan
Students: Sandesh Borgaonkar, Arnab Das

Introduction

We propose to implement an interval branch and bound algorithm(BB) that aims to find the global maxima for a function in a given interval. The goal is to progressively find the best solution for the given cost function using branch and bound technique until a termination condition is reached. Branch and bound techniques deal with the optimization problems over a search space that can be presented as the leaves of a search tree. BB is guaranteed to find an optimal solution, but its complexity in the worst case is as high as that of exhaustive search. However, with the combination of interval analysis, it aims to reduce the search space of intervals containing the extrema providing faster convergence. In following sections we discuss the crux idea behind combining interval analysis with branch and bound, possibilities of parallelization and leveraging using ILCS(iterated local champion search) framework.

Interval Branch and Bound

Branch and Bound(BB) is a principal problem solving paradigm for finding solutions to combinatorial optimization problems in many areas, including computer science and operations research, mainly for computationally intensive, NP-hard problems. The underlying idea of BB is to take a given problem that is difficult to solve directly, and decompose it into smaller partial problems in a way that a solution to a partial problem is also a solution to the original problem. This is iteratively applied to a partial problem until it can be solved directly or is proven not to lead to an optimal solution. The interval branch and bound algorithm(IBBA) is basically BB operating in a search space of intervals, wherein the cost function is re-coded using interval arithmetic.

Consider $x=(x_1, x_2, ..., x_n)$ the real vectors and $X=(X_1, X_2,..., X_n)$ the interval vectors. An interval function $F: I^n \rightarrow I$ is said to be an interval extension of the real-valued function $f:R^n \rightarrow R$ if $f(x) \epsilon F(X)$, whenever $x \epsilon X$. An interval function is said to be inclusion monotonic if $X \subset Y$ implies $F(X) \subset F(Y)$. Results from Moore[3], states that if F is an inclusion monotonic interval extension of f, then F(X) contains the range of f(x) for all $x \subset X$.

Interval analysis for solving optimization problem relies on Moore's result and use BB to find the optima of the function. Given an initial interval domain, it is divided into sub-intervals(called boxes). This forms the branching part of BB. Next each of these intervals are evaluated over the inclusion function F. Boxes that fail to be in bounds of the best solution are discarded while the surviving boxes are further divided into sub-intervals for evaluation. This continues recursively until a termination condition is reached, that is, the queue containing the valid intervals becomes empty or the interval width has reduced beyond a certain tolerance.

Below is a sample pseudo-code for this algorithm from [1].

```
1: Q ← X_0
2: while Q not empty do
3:    Extract X with highest priority p_X from Q
4:    if upperbound(F(X)) < f_best
      or width(X) ≤ ε_x
      or width(F(X)) ≤ ε_f then
5:       Go to step 3
6:    else
7:       Split X in two sub-boxes X_1 and X_2
8:       for i ∈ {1, 2} do
9:          e_i ← f(midpoint(X_i))
10:         if e_i > f_best then
11:            f_best ← e_i
12:            X_best ← X_i
13:         end if
14:         Insert X_i into Q, with priority e_i
15:      end for
16:   end if
17: end while
18: return (X_best, f_best)
```

Description of the pseudocode:

The initial interval domain X_0 is divided into multiple intervals and kept in a priority queue, Q. The 'while' loop continues until Q becomes empty. For every iteration it dequeues an interval from Q and checks its upperbound and interval width tolerance against the current best solution and the allowed tolerance respectively. If any of them return false, that interval is discarded and a new interval is dequeued. For a valid interval, the interval is further split into two subintervals X_1 and X_2. For each of these sub-intervals, the function is evaluated at the mid-point(just a heuristic) of the interval and compared with the current best solution. If the evaluation gives a better solution than the current best, then the current best is updated and the corresponding subinterval is inserted into the queue, else the interval gets discarded. As evident, we always have a current best solution. Hence, when the termination condition is reached (queue empty), the current best value gives the global optima.

Suitability for gpu acceleration

The pseudo-code suggests an inherently sequential algorithm wherein each dequeued interval is evaluated in sequence and the evaluation on the current best solution depends on the immediate last iterated interval. However, we feel the function evaluation for the intervals currently in the queue provides a significant scope for parallel computation wherein multiple threads could be forked off for the function evaluation across respective intervals currently in the queue and update the shared memory(visible to all) location with the current best value that can be read by all other threads. Once the computation returns and the current best in the iteration has been decided, the queue can be filled with new set of intervals resulting from the current best interval and new computation kernel can be forked again until a termination condition is reached.

Challenges

The primary challenge is to synchronize the current best solution evaluation across the multiple threads of interval evaluation and possible heuristics to not get stuck in local optimas.

Leveraging using ILCS framework

ILCS(Iterated local champion search) framework is a highly scalable parallelization framework for running iterative local searches on heterogeneous HPC platforms. The framework requires single cpu/gpu code which it then executes using MPI between compute nodes and openMp and multi-Gpu support within nodes. It manages the seed distribution, node to node communication and tracking the current best solution until termination is reached.

We wish to leverage our cpu-gpu implementation of the interval branch and bound algorithm using the ILCS framework for execution across multiple nodes with multi-gpu support. The large scale parallelization of the problem will reduce the time complexity of the problem to the order of maximum sequential code in the program.

Additionally, we wish to leverage the verification aspects of this project using our CS-6110 project pertaining to symbolic execution aided test-generation, wherein the generated test input files can be reused for testing correctness of our implementation.

References

[1]Finding and Proving the optimum:Cooperative Stochastic and Deterministic Search – Jean-Marc Alliot, Nicolas Durand, David Gianazza, Jean-Baptiste Gotteland
[2]Introduction to interval analysis – Ole Caprani, Kaj Madsen, Hans Bruun Nielson
[3]Interval Analysis – R.E.Moore
[4]A Scalabale Heterogeneous Parallelization Framework for Iterative Local Searches – Martin Burtscher, Hassan Rabeti