Multi-Dimensional Parallel Genetic Algorithm using OpenMP

Zubia Shahid*

University of Sciences, Philadelphia, PA†

Imran Khan‡

genmatixs.com§

(Dated: August 8, 2014)

Abstract

We present a detailed description of a multi-dimensional genetic algorithm implemented in C++11 and the OpenMP library. The use of the OpenMP library allows for the algorithm to be implemented using parallel processing and hence operate in an extremely efficient manner. The algorithm also offers a considerable degree of flexibility in allowing the user to experiment with different cross-over operators. Results are presented for several multi-dimensional functions and in comparison to more common genetic algorithms available.

Keywords: Genetic Algorithms, Parallel Algorithms, OpenMP

* zshahid@genmatixs.com; zshahid@us.edu

† genmatixs.com

‡ ikhan@genmatixs.com; ikhan@aer.com

§ Atmospheric and Environmental Research Inc

INTRODUCTION

Genetic algorithm(GA) is a type of search algorithm, which is designed to find the optimal solution of the given problem in complex search space. It uses the principal of natural selection and evolution to produce several solutions of the problem. Genetic Algorithm works by removing the weak solution from the search space. It evaluates the fitness of the individual solution by calculating its fitness value depending upon the function provided by the user. The individuals having higher fitness value are then selected, cross-bred and mutated in order to give the best possible solution. This cycle continues until the user gets the satisfied results.

Multi-dimensional parallel genetic algorithm with OpenMP has two distinguishing features which are as follows:

- multi-dimensional genetic algorithm gives flexibility to the user to find solution of the function containing multiple dimensions.

- parallel genetic algorithm uses OpenMP which makes it more faster by reducing the time it takes to run.

In the past most of the genetic algorithm dealt with very restricted number of dimensions. However, this genetic algorithm will be able to perform in multi-dimensional search space to give the best possible solution of complicated problems. Additionally,the use of OpenMp instead of MPI(Message Passing Interface) has also proved to be a better option for parallel computing purpose. It is considered easier to program and debug as compared to MPI. OpenMP allows directives to be added incrementally, so the program can be parallelized one portion after another and thus no dramatic change to code is needed. Whereas, more changes are required in converting from serial to parallel programming in MPI. OpenMP is also considered best for loop parallelization, which has an extensive use in this genetic algorithm.

GENETIC ALGORITHM COMPONENTS

Chromosome Builder

The Chromosome Builder is responsible for creating a family of chromosomes from the domain space chosen by the user. Each chromosome is represented as a dynamic array of *boolean* values. Although C++ provides a bitset data structure, as part of its Standard Template Library (STL), it does not have the flexibility of determining the length of a bitset at runtime. Therefore, an array was chosen which could be dynamically altered (std::vector) to map to the user supplied domain space. The length of the chromosome is bound by the domain space, for a given dimension of range [a:b] and a precision, P we have

$$2^{N_i-1} < (b-a) \cdot 10^P \leq 2^{N_i} \tag{1}$$

where N_i is the apportioned length of the chromosome for a given dimension. The total length of the chromosomes is then,

$$N_{length} = \sum_{i=1}^{N_{dim}} N_i \tag{2}$$

The OpenMP library allows for each dimension to be computed independently of the others, with a master thread combining all the lengths into a total length. The total number of chromosomes can be set by the user with a default value of 75, the initial values of each chromosome are randomly chosen.