Exploitation of the potentiality of graphic cards for optimising phylogenies problem by using evolutionary algorithm

Nour El-Houda BENALIA and NourEddine DJEDI

LESIA Laboratory/ Biskra University
BP 145 RP Biskra, Algeria
benaliahouda@live.fr
noureddine.djedi@lesialab.net

Keywords: Evolutionary Algorithm, Optimization and Acceleration, GPU Computing, Performance Evaluation, Recurrent Neural Network.

1 Introduction

Nowadays, many algorithms are redesigned and rewritten for the GPU since modern GPUs [1] introduce massive parallelism for a budget price and new APIs simplify the development of parallel applications.

The aim of humanoid robotic researchers is to obtain robots that can imitate the human behaviours to collaborate, in the best way, with humans. An obvious problem confronting humanoid robotics [2] is the generation of stable and efficient gaits in a reasonable time. In order to address this problem, alternative, biologically inspired control methods have been proposed, which do not require the specification of reference trajectories. The objective of this work is to propose a model that accelerates a method already proposed [2] which is the combination of evolutionary algorithm and recurrent neural network. Evolutionary algorithms are very effective in solving many practical problems, but their execution time can become a limiting factor for some huge problems [3], because a lot of candidate solutions must be evaluated. The power of these algorithms depends on various factors, including the available computing power. Increasing it is interesting, as it would allow to explore the usually huge search spaces more widely and deeply, for better results. There is variety of possibilities how to accelerate EAs [4].

Modern graphics hardware plays an important role in the area of parallel computing. Used to accelerate gaming and 3D graphics applications [1], they have been recently used to accelerate computations for various topics. In order that the controller of the humanoid robot can emerge toward good solutions, he must take a great time. To overcome this problem, the obvious idea is to accelerate the evolutionary process by immersing the evolutionary algorithm used in GPUs. The purpose of this paper is to propose a technique to accelerate the process of the controller in GPUs. To do this, we proposed four combinations for the acceleration of the controller:

1- A recurrent neural network with an evolutionary algorithm parallelized by the model Master/Slave.
2- A recurrent neural network with an evolutionary algorithm parallelized by the island model.
3- A recurrent neural network with an evolutionary algorithm parallelized by the model Master/slave taking into account the parallelism within chromosome (the parallelism between genes).
4- A recurrent neural network with an evolutionary algorithm parallelized by the island model taking into account the parallelism within the chromosome (the parallelism between genes).

The goal of having all these combinations is to choose the most efficient combination. In this paper, we are interesting with the first combination. The results are compared with the serial algorithm by studying the effect of a number of parameters: (a) differing population sizes, (b) differing number of threads, (c) differing problem sizes, and (d) problems having differing complexities, such as evaluation time and interactions among variables.

2 Description of the used model

2-1 Representation of the population

Population is laid out in main memory of GPU, as a two dimensional N x L matrix such that columns refer to chromosomes and rows corresponds to genes within chromosomes as shown in Figure 1, where N is population size and L is chromosome length.

2-2 the evolutionary algorithm with the RNN

The purpose of the evolutionary algorithm (EA) is to optimize the weights of the neural network which controls the humanoid robot [2]. At start-up, the population's chromosomes are initialized at random. The chromosome length is 1130 genes, with 1 gene per RNN weight. The number of connections represents the number of genes in the chromosome; a floating-point number represents each gene.

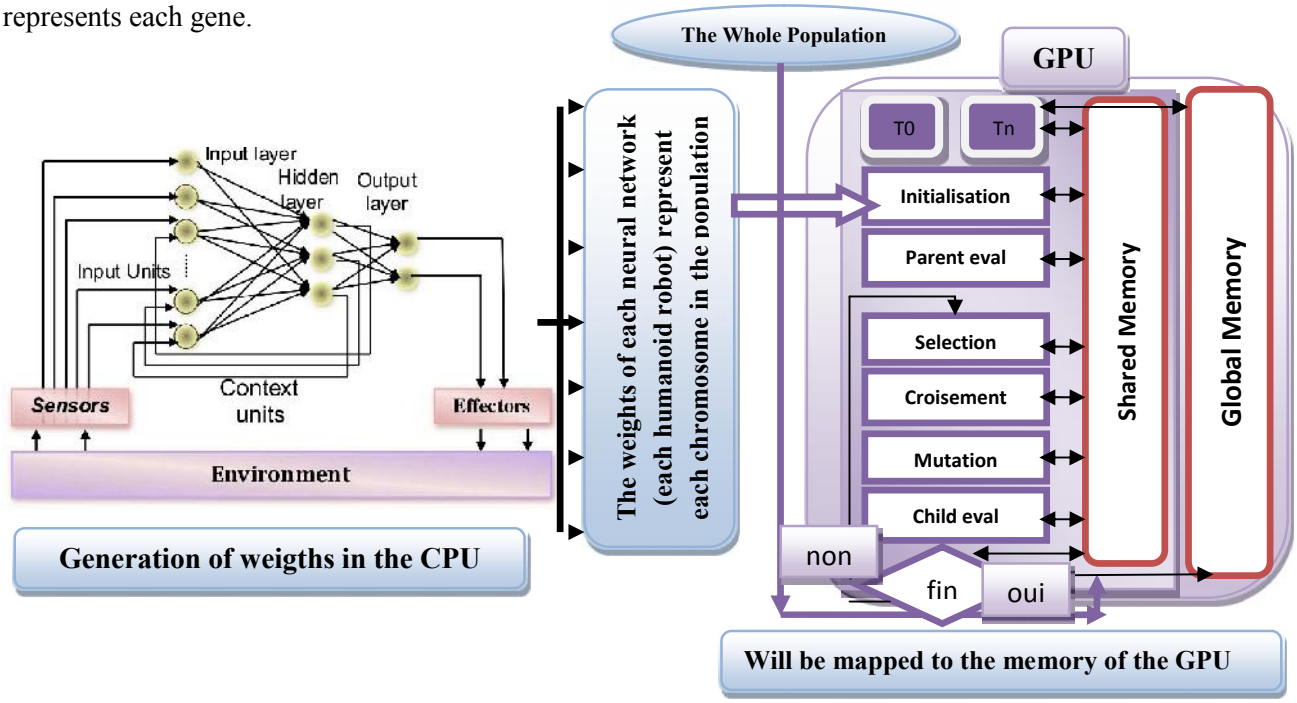

Fig2: the strategy of the combination between the RNN and EA.

The fitness function is based on the distance travelled by the robot within a certain period of time. Our idea is to apply an evolutionary algorithm on the RNN, where we exploit each time a methodology for parallelization of an evolutionary algorithm, as mentioned above. The results of this Implementation will be compared to the sequential one. We are currently implement our method (the first combination) using C/C++ and CUDA (4.0) and run experiments on the Fermi architecture (GTX480). However, this study has to be seen as a first step in the development of a heterogeneous computing solver for a robot controller and application in artificial life.

References

[1] P.Krômer (2011) . An Implementation of Differential Evolution for Independent Tasks Scheduling on GPU.
[2] N.Ouannes (2012). Gait Evolution for Humanoid Robot in a Physically Simulated Environment.
[3] P.Pospichal (2010). Parallel Genetic Algorithm on the CUDA Architecture.
[4] O.Maitre (2012) .EASEA: specification and execution of evolutionary algorithms on GPGPU.
[5] R.Arora (2010). Parallelization of Binary and Real-Coded Genetic Algorithms on CUDA.