Special issue on evolutionary computation on general purpose graphics processing units

José L. Risco-Martín · Juan Lanchares ·
Carlos A. Coello-Coello

Published online: 1 May 2011
© Springer-Verlag 2011

1 Introduction

Evolutionary algorithms (EAs) are metaheuristics inspired by natural phenomena in a broader sense, that have been found to be effective for solving difficult combinatorial optimization problems appearing in various industrial, economical, and scientific domains. Prominent examples for such problems are network design, packing, satisfiability, scheduling, timetabling, transportation, traveling salesperson, vehicle routing and circuits design, to name just a few. Given the large amount of computational resources frequently required for solving hard problems, ideas taken from parallel computing have been successfully applied to different EAs and other bio-inspired metaheuristics during the last few years, including approaches such as Simulated Annealing, Genetic Algorithms, Ant Colony Optimization, Artificial Immune Systems and Genetic Programming, among others.

In this regard, the use of Graphics Processing Units (GPUs) in scientific computing is becoming increasingly common. GPUs are low cost parallel processors that can readily be exploited for many types of general purpose computation. Recently, the computational intelligence

community has started to develop implementations for GPUs platforms. With the advances of modern consumer-level GPUs, parallel EAs can be designed, that fit on Single Instruction, Multiple Data (SIMD)-based GPUs. With the low-cost GPUs currently available in ordinary PCs, more people are becoming able to use parallel evolutionary algorithms to solve the (ordinarily difficult) problems encountered in real-world applications.

The performance of these GPUs is growing at an extraordinary rate. In fact, over the last decade, the processing power of GPUs has been growing at a rate faster than Moore's Law, which governs the performance's growth rate of CPUs. A further improvement in these new GPUs is the increase in pixel depth from 32 bits per pixel to 128 bits per pixel. This means that each red, green, blue, and alpha component can now have a 32-bit floating point accuracy throughout the graphics pipeline. This increase in data accuracy combined with the increased programmability of the GPU means that the GPU is moving towards a more general purpose processor design.

As the guest editors of this special issue, we are pleased to introduce a collection of articles that highlight some of the most interesting aspects of programming EAs in General Purpose GPU (GPGPU) architectures. On the one hand, the selected papers include implementations using diverse GPGPU environments, such as nVIDIA CUDATM, MatLab and AccelerEyes' Jacket toolbox. On the other hand, the selected papers include different types of EAs, such as Genetic Programming (GP), genetic algorithms, particle filter algorithms, memetic algorithms and P systems.

The papers have been selected considering a wide set of readers, including specialists in GPGPU computing, researchers interested in metaheuristics, and those interested in solving practical optimization problems using EAs.

J. L. Risco-Martín (✉) · J. Lanchares
Department of Computer Architecture and Automation,
Universidad Complutense de Madrid, Madrid, Spain
e-mail: jlrisco@dacya.ucm.es

J. Lanchares
e-mail: julandan@dacya.ucm.es

C. A. Coello-Coello
Computer Science Department, CINVESTAV-IPN,
Mexico City, Mexico
e-mail: ccoello@cs.cinvestav.mx

2 Summary of the special issue

Six papers have been carefully selected for publication in this special issue.

Cano, Zafra and Ventura propose an efficient scalable and massively parallel GPU evaluation model to speed up the evaluation phase of GP classification algorithms. The parallel execution model proposed by the authors, together with the computational requirements of the evaluation of individuals, provide an ideal execution environment in which GPUs are a powerful choice. The authors use the nVIDIA CUDA GPU programming model to speed up the fitness calculation phase and reduce the corresponding computational time, reaching a speedup of up to $820\times$.

Contreras, Jiang, Hidalgo and Nuñez-Letamendia present a new multi-objective genetic algorithm to model and optimize automatic trading systems for the stock market. Such algorithm is combined with a GPU-CPU architecture to speed up the power and search capacity of the genetic algorithm for this kind of financial applications. This allows the user to obtain solutions for real-time (or intra-day data) investment decisions. Instead of adopting the nVIDIA CUDA library, the authors use AccelerEyes' Jacket toolbox. Its main advantage is that this toolbox has enough capability to execute Matlab programs in parallel with no specific knowledge of the GPU structure.

Cabido, Montemayor and Pantrigo introduce an approach for visual tracking using GPU for computation. Population-based algorithms have a parallel friendly nature. Thus, they hybridize a particle filter with a memetic algorithm. This gives rise to a new algorithm designed to track single or multiple objects. The authors show that the resulting hybrid algorithm performs better than previous implementations of particle filters. Furthermore, the speed-up obtained using GPUs varies from $5\times$ to $16\times$ for different configurations.

Cecilia, García, Guerrero, Martínez-del-Amor, Pérez-Jiménez and Ujaldón propose membrane systems or P systems as a framework to provide polynomial time solutions to NP-complete problems. Furthermore, the authors exploit the intrinsic parallelism of P systems, as well as a non-intensive floating point nature to develop a GPU model. However, P systems generate an exponential

workspace, and thus, the authors enable different data policies to increase memory bandwidth and exploit data locality through tiling and dynamic queues. To validate the model, they analyze the simulation of a family of recognizer P systems with active membranes that solves the Satisfiability (SAT) problem in linear time on different GPU instances. Furthermore, scalability is illustrated as the authors move to the largest problem size they were able to run.

McKenney and White present a specific GP implementation designed to evolve stock trading strategies using technical analysis indicators. The authors also investigate the speed improvements available when using a GPU for evaluation of individuals in GP, and discuss several issues related to implementing GP on GPUs. In addition, due to the speedups reached through the use of GPU evaluations, the authors investigate the possible improvement in performance when training individuals in a larger number of stocks and training days. It is found that increasing the number of stocks and the length of the training period can result in higher out-of-training testing profitability.

Finally, Maitre, Krüger, Querry, Lachiche and Collet present EAsy Specification of Evolutionary Algorithm (EASEA). EASEA is a software platform that allows the migration of different types of EAs to GPGPUs and clusters of heterogeneous machines using an island model. EASEA can be used to implement genetic algorithms, evolution strategies, memetic algorithms and genetic programming. The authors present the underlying algorithms implemented in EASEA, as well as some results on benchmark and real world problems. Achievable speedups are also shown onto different NVIDIA GPGPUs cards for different optimization algorithm families.

These papers constitute an interesting and representative sample of the current research being conducted on the implementation of evolutionary computation techniques on General Purpose Graphics Processing Units.

Acknowledgments We thank Brunella Gerla, Luis Martínez, and the *Soft Computing* staff for their prompt assistance during the preparation of this special issue. We also thank the anonymous reviewers for their valuable help and for all their hard work. Finally, we are also grateful to all the authors who took the time to submit a manuscript to this special issue.