LITERATURE REVIEW: Parallel Genetic Algorithms on the GPU

George Savin
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
georgesavin@cmail.carleton.ca

October 16, 2022

1 Introduction

Genetic algorithms are powerful metaheuristics that draw insipiration from evolutionary ideas. Through operations such as selection, cross-over and mutation, the solution space is "evolved" and more optimal results are found each iteration (generation). These algorithms are amenable to parallelization and indeed researchers have experimented with parallelizing everything from individual operators to the whole process. However, designing parallel algorithms can differ quite fundamentally given the underlying architecture.

Financial pattern discovery, MAX-3SAT, layout problems, ML hyperparameter selection [] are but a few of the problems GPU GAs have been used to solve.

2 Literature Review

Using GPUs to accelerate Genetic Algorithms started to take off when NVIDIA released CUDA SDK 2.0 in 2008, allowing for more general programming tasks to be parallelized.

A first intuitive approach is to move an operator onto the GPU that can efficiently be ran in parallel, such as the fitness evaluation across a population [9]. Taking this idea further, generation of chromosomes could also moved to the GPU [3]. Unfortunately, this constant transfer between CPU and GPU at every generation slowed down the runtime, and depending on the population size, may not have even been worth it [12].

To overcome this transfer slowdown, master-slave models of a binary and real coded GAs were completely ported to the GPU [4, 2]. Each operator (tournament selection, two-point and single point crossover, bitwise XOR mutation) became separate CUDA kernels. Both the crossover and mutation kernels in these cases suffered from the possibility of having the same chromosome operated on many times, causing inefficient usage and propagation of sub-optimal solutions.

Because of the architectural and operational constraints of the GPU, such as limited shared memory and expensive global memory lookups, models that aimed to reduce global communication fared better. Inversely, these models seemed to have less overall accuracy [17], while other works found this difference non-existant [5]. Further follow-ups to quality of solutions between models have not been explored.

Island model GAs divide populations in the hope that diversity is preserved during evolution. On a GPU, this roughly translates to thread blocks as islands, and single threads to individuals. Within a block, fast memory access and synchronisation is available. Migration between islands occurs asynchronously. Early island model implementations [11, 16] used global memory only for migration. They were applied to numerical and combinatorial optimisations problems with tremendous speed-ups recorded. These speed-ups unfortunately were due mainly to comparisons with poorly optimized sequential GAs [7]. When compared against properly parallelized GAs on CPUs, the speed-up was more in line with expected theoretical analysis. A similar revision of speed-up was seen with newer master-slave implementations [14]. Trade-offs between speed and solution quality were analyzed and associated to parameter tuning, specifically island, generation and chromosome counts [15].

During this time, research looking at optimizing GA GPU representations as well as technique improvements to leverage more parallelization was taking off. Building on the island model implementations, simulated annealing was shown to provide faster convergence when replacing mutation [8]. Memory layouts were also explored, and chromosome based layouts proved to increase locality and make better usage of caches [5]. Different encoding representations also increased convergence speed at no detriment to solution quality [10].

Newer work incorporates some of these techniques, as well as developing new ones targeting Island models almost exclusively. Random seed improvement lead to a solution technique that only generates a single random seed, yet still benefits from the uniqueness and speed one would typically get by generating seeds every generation [15]. Another recent paper used the idea of synchronous migration intervals to improve solution quality by avoiding unintended migrations [6]. This one in particular also used the idea of allocating multiple threads per individual [13] combined with better data organization and found that their techniques provided up to 18x speedups and better solution quality compared to the original IMGAs on the same hardware. Newer work tries to use warp granularity to represent each island and reduce thread divergence [1]. They also perform synchronous and asynchronous replacement, improving solution quality, dubbing it Two-Replacement Policy. Unfortunately, these interesting island model changes were not compared to any prior island model implementations.

References

[1] Faiza Amin and Jinlong Li. Two-Replacements policy island model on GPU. In *Advances in Swarm Intelligence*, pages 242–253. Springer International Publishing, 2022.

[2] Ramnik Arora, Rupesh Tulshyan, and Kalyanmoy Deb. Parallelization of binary and real-coded genetic algorithms on GPU using CUDA. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[3] Stefano Cavuoti, Mauro Garofalo, Massimo Brescia, Antonio Pescape', Giuseppe Longo, and Giorgio Ventre. Genetic algorithm modeling with GPU parallel computing technology. In Bruno Apolloni, Simone Bassis, Anna Esposito, and Francesco Carlo Morabito, editors, *Neural Nets and Surroundings: 22nd Italian Workshop on Neural Nets, WIRN 2012, May 17-19, Vietri sul Mare, Salerno, Italy*, pages 29–39. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] Debattisti, Marlat, Mussi, and others. Implementation of a simple genetic algorithm within the cuda architecture. *The Genetic and*, 2009.

[5] Paul Jähne. Overview of the current state of research on parallelisation of evolutionary algorithms on graphic cards. *Informatik 2016*, 2016.

[6] Dylan M Janssen, Wayne Pullan, and Alan Wee-Chung Liew. Graphics processing unit acceleration of the island model genetic algorithm using the CUDA programming platform. *Concurr. Comput.*, 34(2), January 2022.

[7] Jiri Jaros and Petr Pospichal. A fair comparison of modern CPUs and GPUs running the genetic algorithm under the knapsack benchmark. In *Applications of Evolutionary Computation*, pages 426–435. Springer Berlin Heidelberg, 2012.

[8] Cheng-Chieh Li, Chu-Hsing Lin, and Jung-Chun Liu. Parallel genetic algorithms on the graphics processing units using island model and simulated annealing. *Advances in Mechanical Engineering*, 9(7):1687814017707413, 2017.

[9] Ogier Maitre, Laurent A Baumes, Nicolas Lachiche, Avelino Corma, and Pierre Collet. Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1403–1410, New York, NY, USA, July 2009. Association for Computing Machinery.

[10] Martín Pedemonte, Enrique Alba, and Francisco Luna. Bitwise operations for GPU implementation of genetic algorithms. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, GECCO '11, pages 439–446, New York, NY, USA, July 2011. Association for Computing Machinery.

[11] Petr Pospichal, Jiri Jaros, and Josef Schwarz. Parallel genetic algorithm on the CUDA architecture. In *Applications of Evolutionary Computation*, pages 442–451. Springer Berlin Heidelberg, 2010.

[12] Denis Robilliard, Virginie Marion-Poty, and Cyril Fonlupt. Population parallel gp on the g80 gpu. In *European Conference on Genetic Programming*, pages 98–109. Springer, 2008.

[13] Rajvi Shah, P Narayanan, and Kishore Kothapalli. Gpu-accelerated genetic algorithms. *cvit. iiit. ac. in*, 2010.

[14] Rashmi Sharan Sinha, Satvir Singh, Sarabjeet Singh, and Vijay Kumar Banga. Speedup genetic algorithm using C-CUDA. In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 1355–1359, April 2015.

[15] Xue Sun, Ping Chou, Chao-Chin Wu, and Liang-Rui Chen. Quality-Oriented study on mapping island model genetic algorithm onto CUDA GPU. *Symmetry*, 11(3):318, March 2019.

[16] Thé Van Luong, Nouredine Melab, and El-Ghazali Talbi. GPU-based island model for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 1089–1096, New York, NY, USA, July 2010. Association for Computing Machinery.

[17] Long Zheng, Yanchao Lu, Mengwei Ding, Yao Shen, Minyi Guoz, and Song Guo. Architecture-based performance evaluation of genetic algorithms on Multi/Many-core systems. In *2011 14th IEEE International Conference on Computational Science and Engineering*, pages 321–334, August 2011.