Preparación de Artículos revista VISIÓN ELECTRÓNICA: algo más que un estado sólido
Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

A RESEARCH VISION

QUADRATIC ASSIGNMENT PROBLEM (QAP) ON GPU THROUGH A MASTER-SLAVE PGA

PROBLEMA DE ASIGNACIÓN QUADRÁTICA (PAC) SOBRE GPU A TRAVÉS DE UNA PGA MAESTRO-ESCLAVO

Julián Octavio Castellanos Millán[1] **Víctor Hugo Amarillo Calvo**[2] **Roberto Manuel Poveda Chaves**[3]

Abstract

This document describes the implementation of a Master–Slave Parallel Genetic Algorithm (PGA) on Graphic Processing Units (GPU) to find solutions or solutions close- to optimal solutions to particular instances of the Quadratic Assignment Problem (QAP). The efficiency of the algorithm is tested on a set of QAPLIB standard library problems.

Keywords: Parallel Genetic Algorithms (PGA), Compute Unified Device Architecture (CUDA), Quadratic Assignment Problem (QAP), Graphics Processing Unit (GPU).

1 BSc. In Systems, Universidad Distrital Francisco José de Caldas, Colombia. Current position: Junior Development in informatics and Technology Stefanini Colombia S.A. E-mail: julcas.ud@gmail.com

2 BSc. In Systems, Universidad Distrital Francisco José de Caldas, Colombia. Current position: Development Team Manager in ALDEAMO. E-mail: torvic87@gmail.com

3 BSc. In Mathematics, MSc. In Systems, PhD (c) In Systems and Computation Engineering, Universidad Nacional de Colombia. Current Position: Professor Universidad Distrital Francisco José de Caldas, Colombia. E-mail: rpoveda@udistrital.edu.co

Preparación de Artículos revista VISIÓN ELECTRÓNICA: algo más que un estado sólido
Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

A RESEARCH VISION

Resumen

Este documento describe la implementación de un algoritmo genético paralelo maestro-esclavo (AGP) en unidades de procesamiento gráfico (UPG) para encontrar soluciones -o soluciones cercanas a soluciones óptimas para casos particulares del Problema de asignación Cuadrática (PAC). La eficiencia del algoritmo se prueba en un conjunto de problemas de la biblioteca estándar QAPLIB.

Palabras clave: Algoritmos genéticos paralelos (AGP), Cálculo de Arquitectura Unificada de Dispositivos (CAUD), Problema de Asignación Cuadrática (PAC), Unidad de Procesamiento gráfico (UPG).

1. Introduction

The QAP is a combinatorial problem introduced by Koopmans and Beckmann in 1957 [1] and belongs to the NP-complete problems [2], the QAP consists of finding the optimal assignment of facilities to locations, knowing the distances between cities and the flow between locations. This problem is considered one of the most complex combinatorial optimization problems and it is the model of many problems in the real world, like facilities location, fields planning, computer keywords design, electronic keyboard cabling and other areas [3].

Diverse exact methods have been used, like Branch and Bound, Dynamic Programming, or Back-tracking to solve this problem; [4] shows outstanding research applying these methods to instances with a relatively small problem size; to this day there is no exact algorithm that can solve in a reasonable computational time problems of $n > 40$. QAPLIB library [5] presents a great quantity of QAP test problems of a given size along with the better known solutions.

Metaheuristics methods have emerged in the last 20 years to approach the problem; in particular Evolutionary Computation, as a robust and flexible alternative to solve these

complex optimization problems; modern hardware architectures like GPU offer the possibility to execute those algorithms in parallel, thus diminishing execution time significantly.

2. Preliminaries

2.1. Quadratic Assignment Problem QAP

QAP consists of assigning a set of n facilities in a set of locations; cost is a function of the distance between locations and the flow between facilities. The objective is to assign each facility to each location so that the cost is minimized.

The mathematical model (that corresponds to the Koopmans & Beckmann's original formulation) is (1):

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\sigma(i)\sigma(j)} \tag{1}$$

Where $D = (d_{kl})$ is a distance matrix, $F = (f_{ij})$ is a flow matrix (D and F both size $n \times n$) y $S_n = \{\sigma | \sigma : N \to N\}$, where $N = \{1,2, \dots, n\}$ (it is often said that n is the QAP size). Each individual product $f_{ij} d_{\sigma(i)\sigma(j)}$ of the previous formula is the cost to assign facility i to location $\sigma(i)$ and facility j to location $\sigma(j)$.

.

Sahi & Gonzalez—besides showing that the QAP is NP-complete—show that it is impossible to find an approximate solution in polynomial time within any constant factor of the optimal solution unless [2].

Another reason that highlights the QAP complexity is that many other outstanding NP-complete problems like LAP (Linear Arrangement Problem), GPP (Graph Partitioning

Problem), MCP (Maximum Clique Problem), PPG (Packing Problems in Graphs) and TSP (Travelling Salesman Problem) are just particular QAP instances [6]. For instance:

$$TSP: \min_{\sigma \epsilon S_n} \sum_{i=1}^{n-1} d_{\sigma(i)\sigma(i+1)} + d_{\sigma(n)\sigma(1)} \equiv \min_{\sigma \epsilon S_n} \sum_{i=1}^{n-1} f_{i(i+1)} d_{\sigma(i)\sigma(i+1)} + f_{n1} d_{\sigma(n)\sigma(1)}$$

$$\equiv \min_{\sigma \epsilon S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\sigma(i)\sigma(j)} : QAP \quad (2)$$

Taking for the QAP a $D = (d_{ij})$ as the TSP matrix distances and a $F = (f_{ij})$ as an adjacency matrix of a n vertices Hamiltonian cycle, F can be the matrix:

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}$$

2.2. Genetic Algorithms

Genetic Algorithms (GA) is one of the most outstanding approaches in the field of evolutionary algorithms and are defined as general purpose iterative adaptive search procedures with the advantage that they describe in an abstract and rigorous way the collective adaptation of a population of individuals to a given environment, based on a behavior similar to a natural system [7].

The following pseudo code in algorithm 1 summarizes a simple AG:

Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

Algorithm 1. Simple Genetic Algorithm

To produce an initial population of individuals.

***CYCLE while** the termination condition cannot be found.*

Evaluate the aptitude of all individuals.

Select individuals apt for reproduction.

Produce new individuals.

Generate a new population inserting some new individuals and discarding some bad old individuals.

Mutate some individuals.

*Final **CYCLE while***

2.2.1. Parallel Evolutionary Algorithms

Parallel Evolutionary Algorithms come to existence in a natural manner, since each individual is considered an independent unit [8]. Parallel Evolutionary Algorithms, in essence, have the same function as traditional genetic algorithms, but ease problem solving by distributing workloads and operating in a simultaneous manner on the domain of the problem by allowing an agile solution with respect to time and computational effort [9].

There are different types of Parallel Evolutionary Algorithms; they are classified according to the way population individuals interact and of how their size is defined; the master-slave, fine grain and coarse grain models are outstanding [9], this document uses the master-slave model to implement a QAP solution.

- **Master–Slave model (Global parallelization model)**

This model parallelizes an Evolutionary Algorithm in the level of its fitness function. A master process distributes multiple individuals to various slave processes; they return right away the fitness of those individuals. This procedure results efficient given that the evaluation of the fitness function is the task within an Evolutionary Algorithm that consumes a greater processing time, (Figure 1).

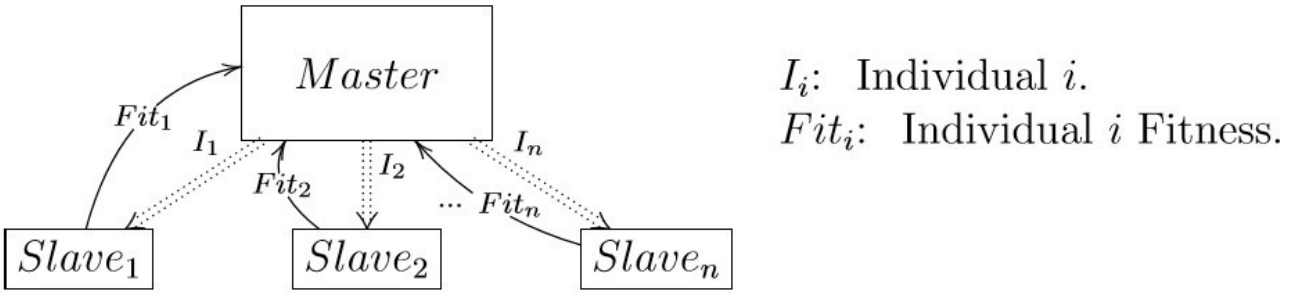

Figure 1. Master-Slave Model. Source: own.

2.3. GPU's

Parallel multiprocessing architectures like Graphic Processing Units (GPU) [10] have significantly evolved in the last 7 years, to increase the graphic processing capabilities in the game industry, to make them faster and more realistic. Such multiprocessing have been used in science for the solution of problems in the real world (computational biology, cryptography, among others) with the help of APIs like CUDA (Compute Unified Device Architecture), OPEN CL, or Direct Compute that exploit those GPU advantages. So, now the term GPGPU is well known (General Purpose Graphic Processing Units).

The main advantage of a GPU is its structure— each GPU contains up to hundreds of cores grouped in multiprocessors of SIMD architecture (one instruction, multiple data).

Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

Evolutionary algorithms are inherently parallel in nature, so they are favored to be implemented on GPU, but considering the challenge of how to adequately handle the access to the device memory.

The QAP has been little studied with parallel models, evolutionary techniques and parallel multiprocessing at the same time. The most important work in this field are:

In [11] they combine evolutionary computation and local search to solve QAPs, the authors emphasize the fact that, to the day, this hybrid implementation had been scarcely used. This work maps the search space in the GPU memory hierarchy minimizing GPU-CPU data transfer; texture memory is dedicated to aptitude function evaluation. The Authors leave the door open to consider multi-GPU architectures that work with a parallel GPU search algorithm.

In [12] the authors use Evolutionary Algorithms to solve the QAP. They implement a Cellular Evolutionary or fine grain Algorithm on a *n*-dimensional toroidal grid as an extension of the usual bi-dimensional grids.

In [13] they use metaheuristics but based on trajectories such as TS (Tabu Search), SA (Simeled Annealing) and PSO (Particle Swarm Optimization) along with 3-opt, 3-opt greedy and VZN to solve the QAP. The authors show that the combination of 3-opt and TS offers the best results to problems in QAPLIB of 12 to 30 nodes.

In [14] they use ACO to solve the QAP. They combine ACO and an efficient method to handle 2-opt heuristics properly assigning CUDA threads; the authors use only one GPU but make it clear that it would be useful to use more than one and propose to use the same algorithm but extended to a Tabu Search.

Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

3. Implementation

An AG global population of 15360 is initialized in the CPU memory and is sent to the GPU device memory so that each thread (256 in total), distributed in 2 blocks, process a fraction of it. Each global population fraction is called subpopulation, and each is composed of 60 individuals where, on each, an AG is executed.

A Roulette Selection is executed [15], a generational Partially Matched Crossover (PMX) [16] with a probability rate of 0,25, and a Simple Inversion Mutation (SIM) [16] with a probability rate of 0,05 (except for subpopulations 0 and 128, they have a mutation rate of 0,6 to give that algorithm a greater level of diversity); tests were made with other selection, crossover and mutation techniques, being the aforementioned the ones that gave the best results.

Each thread makes on each subpopulation a given number of iterations (5, 7, 10, 13, or 15) then, the host examines the general termination condition for all threads. If the condition is satisfied the algorithm ends, otherwise, it is considered that individuals migration phase between subpopulations according to a ring topology, more precisely 5% of the i th subpopulation individuals replace the 5% of individuals of the subpopulation handled by the thread $(i+1)(mod256)$, ($\forall i, 0 \leq i \leq 255$); in each previous 5% there is also a 0.8 probability to include the best individual of the *i* th population to the $(i+1)(mod256)$. The figure 2 describes the implementation.

Fecha de envío: 5 de octubre de 2016
Fecha de recepción: 5 de noviembre de 2016
Fecha de aceptación: 15 de Noviembre de 2016

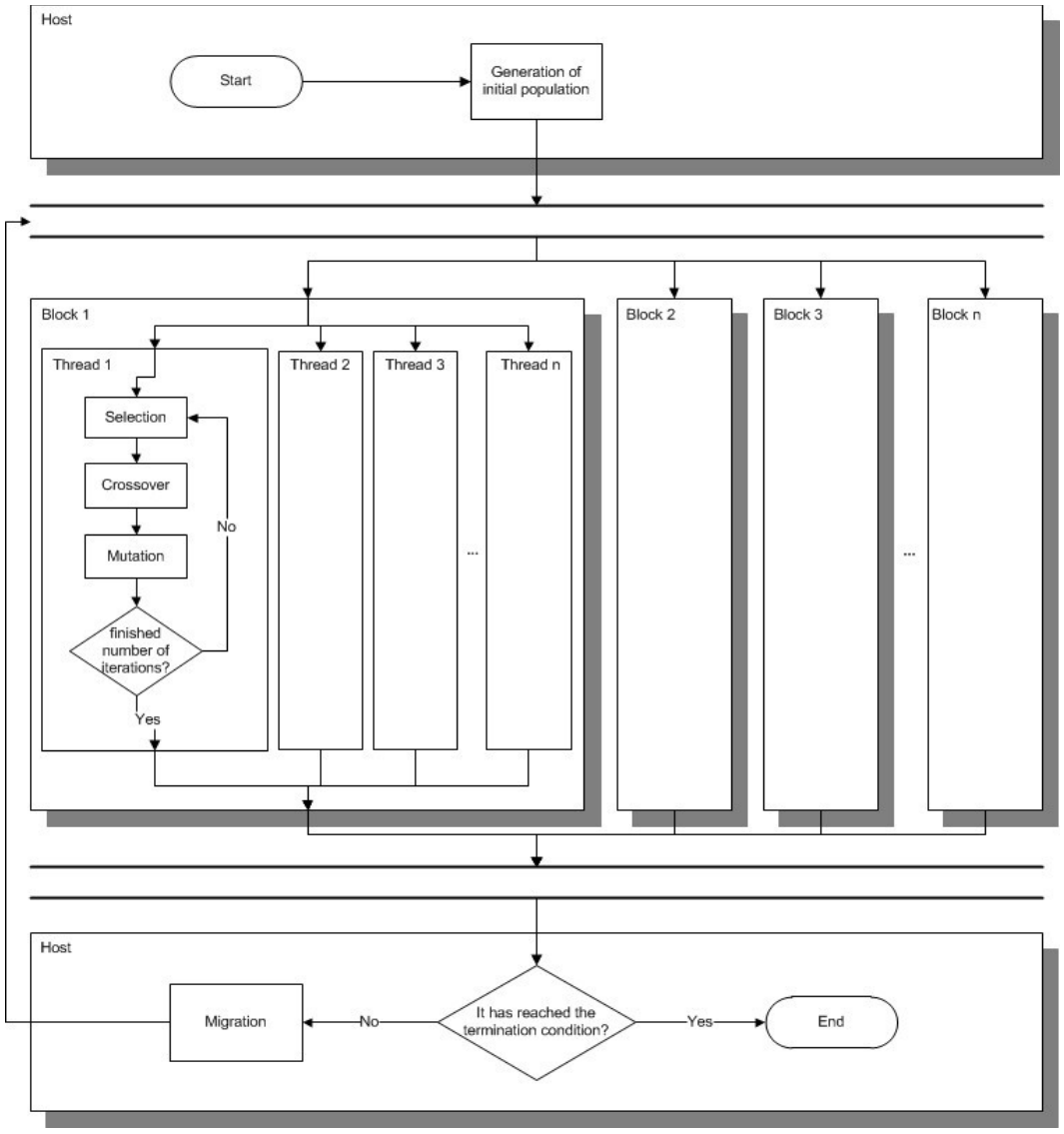

Figure 2: Flowchart PGA Implementation. Source: own.

4. Results and conclusions

The following results were obtained for 13 problems of the size $n = 16$ and two problems size $n = 20$ included in the QAPLIB library. In table 1 we can see the best results found for each problem, with the number of iterations that run the genetic algorithm in each subpopulation

(only the best result is shown, be it in iteration 5, 7, 10, 13, or 15), the quantity of executions controlled by the host and the gap between the found solution and the best reported solution in QAPLIB, according to the formula (3):

$$gap = \frac{\text{Best solution found} - \text{Best solution reported in QAPLIB}}{\text{Best solution reported in QAPLIB}} \quad (3)$$

Name instance	Size of instance	Solution QAPLIB	Iteration of algorithm genetic	Executions in host	Solution found	gap
Esc16a	16	68	5	5	68	0
Esc16b	16	292	5	1	292	0
Esc16c	16	160	5	5	160	0
Esc16d	16	16	5	3	16	0
Esc16e	16	28	5	5	28	0
Esc16f	16	0	5	1	0	0
Esc16g	16	26	10	3	26	0
Esc16h	16	996	7	2	996	0
Esc16i	16	14	5	2	14	0
Esc16j	16	8	5	6	8	0
Had16	16	3720	5	31	3720	0
Nug16a	16	1610	7	83	1610	0
Nug16b	16	1240	5	95	1240	0
Nug20	20	2570	13	122	2614	1,71
Had20	20	6922	10	154	6926	0,06

Table 1: Obtained results. Source: own.

The implementation of the algorithm only considered the local memory of the threads and the global memory of the device; making use of other types of memory that offer GPU (shared memory, texture memory) and with a combination of them there could surely have been better results in a lesser number of iterations (executions).

It can be concluded that using a fine grain or coarse grain parallel model or a hybrid between the two (as Grisland did [17]), it is possible to tackle problems greater that $n = 16$ or $n = 20$, and the use of local search heuristics would surely help find good results for these greater instances of the problem.

References

[1] T. Koopmans and M. Beckman, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, pp. 53-76, 1957.

[2] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the Association for Computing Machinery,* vol. 23, pp. 555--565, 1976.

[3] R. Burkard, E. Cela, P. Pardalos and L. Pitsoulis, "The quadratic assignment problem.," Handbook of Combinatorial Optimization, vol. 2, pp. 241-338, 1998.

[4] E. Loiola, N. de Abreo, P. Boaventura-Nett, P. Hahn and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operation Resarch,* vol. 176, pp. 657-690, 2007.

[5] R. Burkard, Ç. E., S. Karisch and R. F., "QAPLIB - "A Quadratic Assignment Problem Library," Septiembre 2015 [Online]. Available: http://www.opt.math.tu-graz.ac.at/qaplib/

[6] R. Burkard, "The Quadratic Assignment Problem," Handbook of Combinatorial Optimization, pp.2741-2814, 2 nd Edition. Springer: New York, 2013.

[7] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs," Springer-Verlag, 1994.

[8] K. De Jong, W. Spears and D. Gordon, "Using genetic algorithms for concept learning.," *Machine Learning,* vol. 13, no. 2, pp. 161-188, 1993.

[9] M. Tomassini, "A survey of genetic algorithms.," *Annual Reviews of Computational*

Physics, World Scientific, vol. 3, pp. 87-118, 1995.

[10] Nvidia, "CUDA GPUs" Agosto 2015 [Online]. Available: https://developer.nvidia.com/cuda-gpus

[11] T. Luong, N. Melab and E. Talbi, "Gpu-based island model for evolutionary algorithms," Genetic and Evolutionary Computation Conference*,* pp. 1089-1096, 2010.

[12] N. Soca, J. Blengio, M. Pedemonte and P. Ezzatti, "Pugace, a cellular evolutionary algorithm framework on gpus," *IEEE Congress on Evolutionary Computation,* pp. 3891-3898, 2010.

[13] M. Bashiri, "An analytical comparison to heuristic and meta-heuristic solution". Computers and Industrial Engineering (CIE), 2010 40th International Conference on

[14] S. Tsutsui and N. Fujimoto, "An analytical study of gpu computation for solving qaps by parallel evolutionary computation with independent run," *IEEE,* 2010.

[15] L. Thiele and T. Blickle, "A comparison of selection schemes used in genetic algorithms," TIK Report of Swiss Federal Insititute of Technology*,* 1995.

[16] C. Kuijpers, P. Larrañaga, R. Murga, I. Inza and S. Dizdarevic, "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators," *Artificial Intelligence Review,* vol. 13, no. 2, pp. 129-170, 1999.

[17] E. León, J. Gómez and R. Poveda, "Grisland: A parallel genetic algorithm for finding near optimal," *GECCO,* 2009.