UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido

http: revistas.udistrital.edu.co/ojs/index.php/visele/index

VISIÓN ELECTRÓNICA

A RESEARCH VISION

Combinatorial optimization NP-Hard problem solved by using the quadratic assignment problem (QAP) solution through a parallel genetic algorithm on GPU

Problemas de optmización combinatorial NP-Hard solucionados a partir del problema de asignación cuadrática (QAP) solución através de un algoritmo genético paralelo sobre GPU

Eduardo Cárdenas Gómez[1], ***Roberto Poveda Chaves***[2] ***Orlando García Hurtado***[3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 25/09/2016
Recibido: 12/03/2017
Aceptado: 02/09/2017

Keywords:
Combinatorial Optimization
NP-Hard Problem
Compute Unified Device Architecture (CUDA)
Graphics Processing Unit (GPU)
Parallel Genetic Algorithms (PGA)
Quadratic Assignment Problem

Palabras clave:
Problemas de Optimización Combinatorial NP–Difíciles
Arquitectura Unificada de Dispositivos de Cómputo (CUDA)
Unidades de Procesamiento Gráfico (GPU)
Algoritmos Genéticos Paralelos
Problema de Asignación Cuadrática (QAP)

ABSTRACT

In this article, some instances of well known combinatorial optimization NP-Hard problems are solved by using Koopmans and Beckmann formulation of the quadratic assignment problem (QAP). These instances are solved by using an Embarrassingly Parallel Genetic Algorithm or by using an Island Parallel Genetic Algorithm; in both cases, the implementation is carried out on Graphics Processing Units (GPUs).

RESUMEN

En este documento se resuelven algunas instancias de problemas bien conocidos de optimización combinatorial de tipo NP-Hard a partir de la formulación de Koopmans y Beckmann del problema de Asignación Cuadrática (QAP). Dichas instancias son solucionadas mediante un Algoritmo Genético Embarasosamente Paralelo o mediante un Algoritmo Genético Paralelo de Islas, en ambos casos, la implementación se hace sobre unidades de procesamiento gráfico (GPU's).

[1] BSc. In Mathematics, Universidad Nacional de Colombia, Colombia. MSc. in Systems Engineer, Universidad Nacional de Colombia, Colombia. Current position: Professor at Universidad Nacional de Colombia,. E-mail: ecardenasg@unal.edu.co.

[2] BSc. In Mathematics, Universidad Nacional de Colombia. MSc. in Systems Engineer, Universidad Nacional de Colombia. PhD (c). in Systems Engineer, Universidad Nacional de Colombia Current position: Professor at Universidad Distrital Francisco José de Caldas, Colombia. E-mail: rpovedac@udistrital.edu.co.

[3] BSc. In Licensed in Mathematics, Universidad Distrital Francisco José de Caldas, Colombia. PhD (c). in Mathematics Education, Universidad Antonio Nariño, Colombia. Current position: Professor at Universidad Distrital Francisco José de Caldas, Colombia. E-mail: ogarcia68@gmail.com.

Citar este artículo como: E. Cárdenas G., R. Poveda C., O. García H. "Combinatorial optimization NP-Hard problem solved by using the quadratic assignment problem (QAP) solution through a parallel genetic algorithm on GPU". Visión Electrónica, algo más que un estado sólido, Vol. 11, No. 2, 146-151, july-december 2017. https://doi.org/10.14483/22484728.13130

1. Introduction

Some NP-Hard type combinatorial optimization problems end up being particular instances of the Koopmans and Beckmann Quadratic Assignment Problem [0].

The QAP consists of finding the optimal assignment of facilities to n locations, knowing the n distances between facilities and the flow between locations. The QAP is considered a 'strongly NP-Hard' problem. Sahni and Gonzalez (1976) show that this problem is not only an NP-Hard problem but that it is also impossible to find a solution by using an ϵ-approximate polynomial algorithm unless $P = NP$. To this day, there is no exact algorithm that can solve in a reasonable computational time problems of size $n > 40$. The Traveling Salesman Problem is an approximation in polynomial time within $\epsilon = 3/2$ in the case that the distance matrix is symmetric and satisfies the triangle inequality [2]. Metaheuristics methods have emerged in the last 20 years to approach the problem; in particular Genetic Algorithms, 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 significantly execution time.

2. Preliminaries

2.1. *Quadratic Assignment Problem (QAP)*

QAP consists of assigning a set of n facilities in a set of n 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:

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

where $D = (d_{kl})$ is a distance matrix, $F(f_{ij})$ is a flow matrix (D and F both size $n \times n$) and $S_n = \{\sigma | \sigma : N \to N\}$, where $N = \{1, 2, \ldots, n\}$ (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 $\sigma(j)$.

2.2. *Combinatorial Optimization Problems Formulated as QAPs*

Many outstanding NP-hard problems like LAP (Linear Arrangement Problem), MCP (Maximum Clique Problem) and TSP (Travelling Salesman Problem) are just particular QAP instances [3]. In fact:

2.2.1. *The Traveling Salesman Problem (TSP) [4]*

This problem can be stated as follows: How an agent should visit a set of cities returning to the starting city in such a way that each city is just visited once and the cost of the tour is the minimum?

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:

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

$$\begin{aligned} TSP : & \left(min_{\sigma \in S_n} \sum_{i=1}^{n-1} d_{\sigma(i)\sigma(i+1)} \right) + d_{\sigma(n)\sigma(1)} \\ = & \left(min_{\sigma \in S_n} \sum_{i=1}^{n-1} f_{i(i+1)d_{\sigma(i)\sigma(i+1)}} \right) + f_{n1} d_{\sigma(n)\sigma(1)} \\ = & \; min_{\sigma \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\sigma(i)\sigma(j)} : QAP \end{aligned} \tag{2}$$

2.2.2. *The Linear Arrangement Problem LAP [5]*

This problem consists in finding an ordering of the nodes of a weighted graph on $n\}$ nodes, such that the sum of the weighted edge lengths is minimized.

Taking for the QAP a $D = (d_{ij})$ given by $d_{ij} = |i - j|$, for all i, j the flow matrix $F = (f_{ij})$ is the (weighted) adjacency matrix of the given graph $G = (V, E)$.

$$\begin{aligned} LAP : min_{\sigma \in S_n} & \sum_{(i,j) \in E} w_{ij} |\sigma(i) - \sigma(j)| \\ = \; min_{\sigma \in S_n} & \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\sigma(i)\sigma(j)} : QAP \end{aligned} \tag{3}$$

Where w_{ij} is the weight of the edge (i, j).

2.2.3. Maximum Clique Problem MCP [6].

Given a graph G = (V, E) with $|V| = n$, this problem consists in finding the maximum number $k \leq n$ such that there exists a subset $V_1 \subseteq V$ with k vertices which induces a clique in G.

Taking for the QAP a $D = (d_{ij})$ equal to the adjacency matrix of the given graph G, and flow matrix $F = (f_{ij})$ given as adjacency matrix of a graph consisting of a clique of size k and $n-k$ isolated vertices, multiplied by -1. It is easy to show that G has a k-size click if the optimal value of the QAP problem is $-k^2 + k$

2.3. Parallel Genetic Algorithms (PGA)

Parallel Genetic Algorithms come to existence in a natural manner, since each individual (of a simple Genetic Algorithm) is considered an independent unit from the processing viewpoint [7]. Parallel Genetic 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 [8].

There are different types of PGA; they are classified according to the way population individuals interact and of how their size is defined; the embarrassingly parallel , the master-slave, fine grain and coarse grain models are outstanding [8], this document uses the embarrassingly parallel model and coarse grain (*Island Parallel Genetic Algorithm*) to implement a QAP solution.

2.3.1. Embarrassingly Parallel Algorithm (EPA)

The same evolutionary algorithm is run under different initial conditions in a parallel way. When all the different configurations have been executed, the configuration showing the best behavior is chosen.

2.3.2. Island Parallel Genetic Algorithm (Coarse Grain)

The population is divided in several sub-populations of some relative size, each one being evolved in a different processor. Each sub-population is geographically separated from other sub-population. Individuals' migration from one subpopulation to another one is allowed if such sub-populations are neighbors. The migration operation is used in order to introduce diversity in the population and is carried out by following some predetermined patterns. The original Island algorithm is shown in Algorithm 1.

Algorithm 1 Island Model

```
    Produce P subpopulations of size N; generation
    number: = 1
WHILE termination condition not met
    For each subpopulations DO in parallel
    Evaluate and select individuals by fitness
    IF [(generation number) mod (frequency) = 0] then
    Send $K$ ($K < N$) best individuals to a neighboring
    subpopulation
    Receive $K$ individuals from a neighboring
    subpopulation
    Replace $K$ individuals in the subpopulation
    End IF
    Produce new individuals
    Mutate individuals
    End DO in parallel
    Generations number: = generation number + 1
End WHILE.
```

2.4. Graphics Processing Unit (GPU)

Parallel multiprocessing architectures like Graphic Processing Units (GPU) [9] have significantly evolved in the last 9 years, to increase the graphic processing capabilities in the game industry and to make them faster and much more realistic. Such multiprocessing have been used in science for 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. At present, the term (General Purpose Graphic Processing Units) GPGPU is well known.

Additionally, the main advantage of a GPU is its structure, each GPU contains up to hundreds of cores grouped in multiprocessors of SIMD (single instruction, multiple data) architecture. Genetic algorithms are inherently parallel in nature, so they are favored to be implemented on GPU, but considering the challenge of how to handle adequately the access to the device memory.

3. Implementation

Two PGA models (Embarrassingly Parallel and Island Parallel) were implemented independently to solve two different instances of each of the three combinatorial NP-hard problems cited in section 2.2. The chosen instances correspond to prominent benchmark problems found in the literature, they are:

- For the TSP [10]
  - √ burma14 (14 cities in Burma (geographical coordinates)).
  - √ bays29 (29 Cities in Bavaria (street distance)).
- For the LAP [11]
  - √ can_24 (structure symmetric, graph Size: 24), figure 1.
  - √ ibm32 (structure unsymmetric, graph Size: 32), figure 2.
- For the MCP [12]
  - √ MANN_a9 (graph Size:45)
  - √ Johnson8-4-4 (graph Size:28)

Figure 1: can_24 [11].

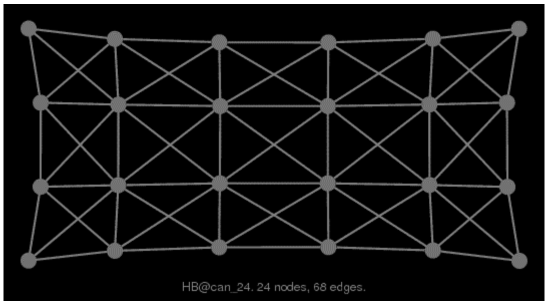
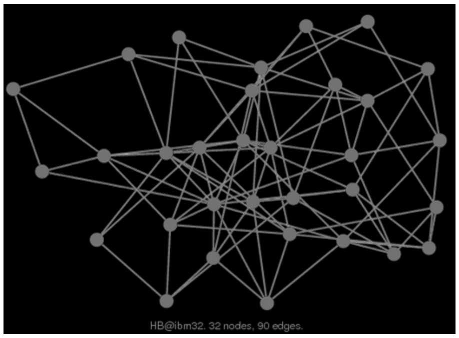
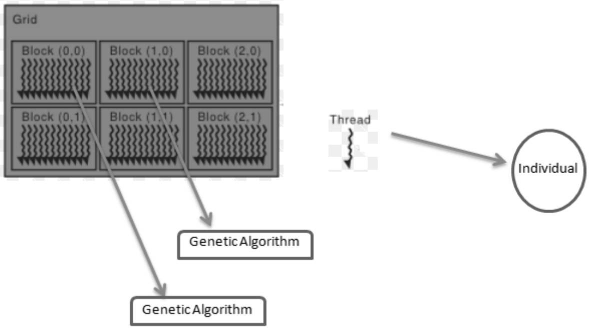


Figure 2: ibm32 [11].



For each PGA model five blocks with 64 GPU threads were used, each thread corresponds to a simple GA individual, and each block corresponds to a subpopulation. In each block a simple elitist GA is executed independently of the other blocks. The Embarrassingly Parallel Algorithm (EPA) implemented is shown in figure 3.

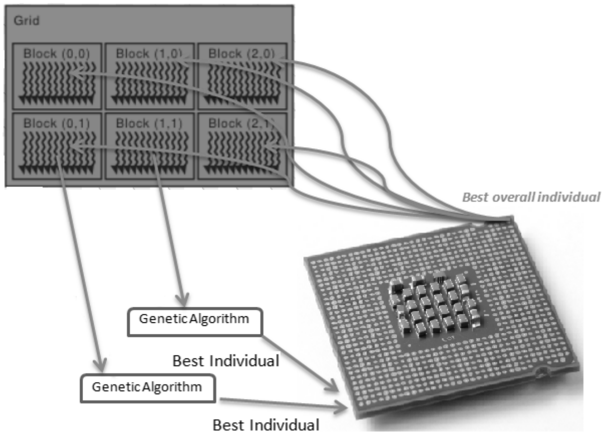
Figure 3: Embarrassingly Parallel model in GPU.



Source: own.

In the Island Genetic Parallel Algorithm, the best individual of each block, after three iterations (iterations in device), is sent to the CPU and the best of all is returned to each GPU block, figure 4.

Figure 4: Island model in GPU.



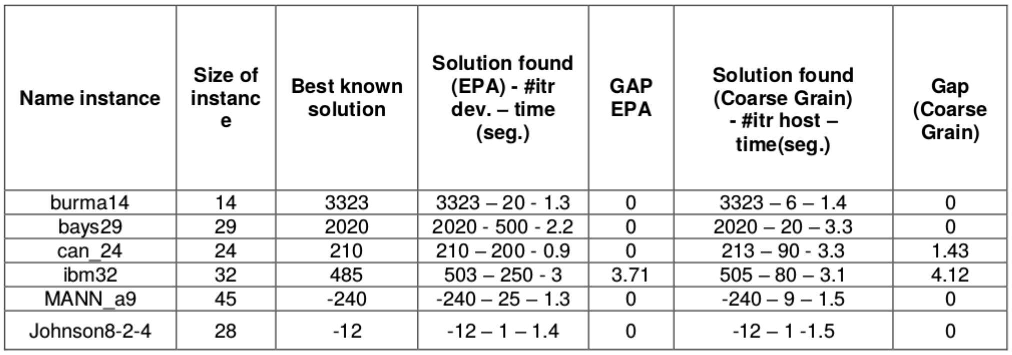
Source: own.

The genetic operators considered in the PGA are the basic selection, crossover, mutation and transposition operators.

A whole coding was used. If A is the chain (chromosome) that represents this coding, then, the position i of the A chain corresponds to the i installation and the $A[i]$ content of the chain (gen $A[i]$) corresponds to the $\sigma(i)$ location.

The selection is Tournament. This method compares individuals of the population (confronting them by the fitness) and choose the most apt, the comparison is made by couples of individuals (a binary tournament).

Table 1: Obtained results.

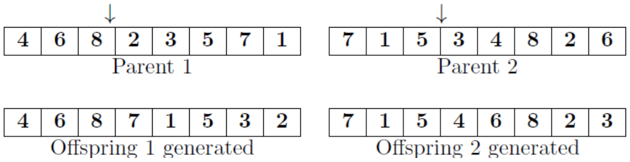


Name instance	Size of instanc e	Best known solution	Solution found (EPA) - #itr dev. – time (seg.)	GAP EPA	Solution found (Coarse Grain) - #itr host – time(seg.)	Gap (Coarse Grain)
burma14	14	3323	3323 – 20 - 1.3	0	3323 – 6 – 1.4	0
bays29	29	2020	2020 - 500 - 2.2	0	2020 – 20 – 3.3	0
can_24	24	210	210 – 200 - 0.9	0	213 – 90 - 3.3	1.43
ibm32	32	485	503 – 250 - 3	3.71	505 – 80 – 3.1	4.12
MANN_a9	45	-240	-240 – 25 – 1.3	0	-240 – 9 – 1.5	0
Johnson8-2-4	28	-12	-12 – 1 – 1.4	0	-12 – 1 -1.5	0

Source: own.

The crossover is a Modified order crossover (MOX). A crossover point common for the two parents is randomly selected, the genes to the left of the crossover point are kept and then the remaining genes of the other parent are copied in the order in which they are, figure 5.

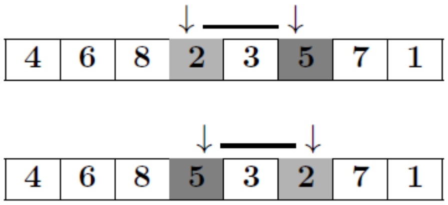
Figure 5: MOX.



Source: own.

The mutation is an Exchange Mutation (EM). This type of mutation selects two genes and exchanges them, figure 6 as follows.

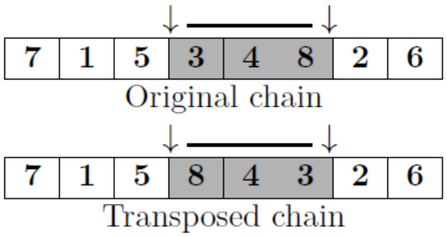
Figure 6: EM.



Source: own.

Transposition simply reverts the chromosome genes between two points randomly generated, as in figure 7.

Figure 7: Transposed.



Source: own.

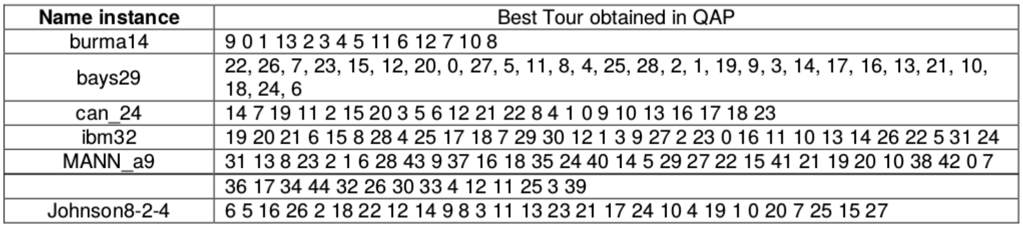
On account of the crossover probability rate is 0.6 in all GPU blocks, and the mutation probability rate is 0.1 in three blocks and 0.9 in the other two; it is important to note that in conjunction with the elitist model used, it is productive to consider a high mutation probability (0.9) in the Island model for all individuals in at least one block, to gain diversity. The rate used in the transposition operator is 0.5 in all blocks.

4. Results and conclusions

In table 1 below, we can see the best solution for each benchmark problem reported by the literature; the best solution found for each one of the two parallel models used (including the quality of iterations in host (#itr) and the time elapsed to reach such response (time)), and the gap between the found solution and the best reported solution, according to the formula:

$$gap = \frac{\text{Best solution found - Best solution repordet in la literatura}}{\text{Best solution reported in la literatura}} \tag{4}$$

Table 2: Best Tour obtained in QAP.



Name instance	Best Tour obtained in QAP
burma14	9 0 1 13 2 3 4 5 11 6 12 7 10 8
bays29	22, 26, 7, 23, 15, 12, 20, 0, 27, 5, 11, 8, 4, 25, 28, 2, 1, 19, 9, 3, 14, 17, 16, 13, 21, 10, 18, 24, 6
can_24	14 7 19 11 2 15 20 3 5 6 12 21 22 8 4 1 0 9 10 13 16 17 18 23
ibm32	19 20 21 6 15 8 28 4 25 17 18 7 29 30 12 1 3 9 27 2 23 0 16 11 10 13 14 26 22 5 31 24
MANN_a9	31 13 8 23 2 1 6 28 43 9 37 16 18 35 24 40 14 5 29 27 22 15 41 21 19 20 10 38 42 0 7
	36 17 34 44 32 26 30 33 4 12 11 25 3 39
Johnson8-2-4	6 5 16 26 2 18 22 12 14 9 8 3 11 13 23 21 17 24 10 4 19 1 0 20 7 25 15 27

Source: own.

The best solution reported by our algorithms is the best solution found in 20 executions for each problem. The algorithms were executed in an Intel® CoreTM i7 - 4700HQ CPU @ 2.40GHz, RAM 8 GB y GPU Nvidia GeForce GTX 760M.

Furthermore, the implementation of the algorithms did not consider the types of constant memory or texture memory, with a combination of them there could surely have been better results in a lesser number of iterations and/or executions.

Consequently, it can be concluded that using a fine grain or coarse grain parallel model or a hybrid between the two, as Grisland did [13] it is possible to tackle greater problems than those considered, in addition to 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, no. 1, pp. 53-76, 1957. pp. 53-76.

[2] N Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem,"Technical Report 338, 1976.

[3] R.E. Burkard, "The Quadratic Assignment Problem," Pardalos P.M. Handbook of Combinatorial Optimization, 2013, https://doi.org/10.1007/978-1-4419-7997-1_22

[4] E Lawler and J Lenstra, "The Traveling Salesman Problem," Wiley, Chichester, 1985.

[5] M Garey and D Johnson, "Computers and Intractability: A Gude to the Theory of NP-Completeness," W.H.Freeman and Company, New York, 1979.

[6] P Pardalos and J Xue, "The maximum clique problem.," *J. Global Optim 4*, pp. 301-328, 1994, https://doi.org/10.1007/BF01098364

[7] K. De Jong, W. Spears, and D. Gordon, "Using genetic algorithms for concept learning.," *Machine Learning*, vol. 13, no. 2, pp. 161-188, 1993, https://doi.org/10.1007/BF00993042

[8] M Tomassini., "A Survey of Genetic Algorithms," *Volume III of Annual Reviews of Computational Physics, World Scientific.*, vol. 3, pp. 87-118, 1995, https://doi.org/10.1142/9789812830647_0003

[9] Nvidia CUDA. May 12 th 2016 [Online]. Available : https://developer.nvidia.com/cuda-gpus

[10] University, Heidelberg. May 12 th 2016 [Online]. Available: http://comopt.ifi.uniheidelberg.de/software/TSPLIB95/

[11] University of Florida The Harn Museum. "The SuiteSparse Matrix Collection" May 12 th 2016 [Online]. Available: http://www.cise.ufl.edu/research/sparse/matrices/

[12] University of Glasgow School of Computing Science. (2012) DIMACS. May 12 th 2016 [Online]. Available: http://www.dcs.gla.ac.uk/~pat/maxClique/distribution/DIMACS_cliques/

[13] E. León, J. Gómez, and R. Poveda, "Grisland: A parallel genetic algorithm for finding near optimal,"GECCO, Montreal, Canada, pp. 2035-2040, 2009.