Applied Mathematical Sciences, Vol. 11, 2017, no. 57, 2843 - 2854
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ams.2017.710319

A Solution for the Quadratic Assignment Problem (QAP) through a Parallel Genetic Algorithm Based Grid on GPU

Eduardo Cárdenas G.[1], Roberto Poveda Ch.[2] and Orlando García H.[3]

[1]Departamento de Matemáticas
Universidad Nacional de Colombia
Bogotá, Colombia

[2]Facultad de Ingeniería
Universidad Distrital "Francisco José de Caldas"
Bogotá Colombia

[3]Facultad de Ingeniería
Universidad Distrital "Francisco José de Caldas"
Bogotá Colombia

Abstract

The QAP is considered one of the most complex combinatorial optimization problems and it is the model for many real life problems such as facilities location, camp planning, computer keyboard design and electronic boards cabling, among other applications. This Paper solves some significant instances of this problem through an improved Parallel Genetic Algorithm based Grid by means of the local heuristics search 2-opt on Graphics Processing Units (GPU's).

Keywords: Combinatorial Optimization NP-Hard Problem, Quadratic Assignment Problem (QAP), Parallel Genetic Algorithms (PGA), Local heuristics search 2-opt, Graphics Processing Unit (GPU), Compute Unified Device Architecture (CUDA)

1 Introduction

The QAP was originally introduced by Koopmans & Beckmann in 1957 [10] and is considered a "strongly NP-Hard" problem. Sahni and Gonzalez [15] 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*.

The problem consists of finding the optimal assignment of n facilities to n locations, knowing the distances between facilities and the flow between locations. To this day there is no exact algorithm that can solve in a reasonable computational time problems of size $n > 40$. 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. Unfortunately, Genetic Algorithms tend to converge prematurely to locally optimal points, so it is necessary to exploit in greater detail promising regions; for this topic, we resort to the local heuristics search 2-opt.

Modern hardware architectures like Graphics Processing Units (GPU's) [4] 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 & Beckmanns original formulation) is:

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

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

The QAP has many different but equivalent mathematical formulations that are the basis for different solutions approaches. The trace formulation of the

QAP is:

$$\min_{X \in X_n} \; trace(FXD^tX^t)$$

where $X = (x_{ij})$ is the permutation matrix associated with a permutation σ

$$x_{ij} = \begin{cases} 1 & \text{if } \sigma(i) = j \\ 0 & \text{otherwise} \end{cases}$$

and X_n is the set of all permutation matrices. This formulation makes better use of the multiprocessing features of the GPU.
Many outstanding NP-hard problems like LAP (Linear Arrangement Problem), MCP (Maximum Clique Problem) and TSP (Travelling Salesman Problem) are just particular QAP instances [2], In fact: for example, in the MCP [13] given a graph $G = (V, E)$ with $|V| = n$, we ask for 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. This problem can be formulated as a QAP taking with distance matrix $D = (d_{ij})$ equal to the adjacency matrix of G, and the 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 $-k^2 + k$.

2.2 Parallel Genetic Algorithms (PGA)

PGA come to existence in a natural manner, since each individual (of a simple Genetic Algorithm) is considered an independent unit from the processing viewpoint [9]. PGA, 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 [17].
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-grained and coarse-grained models are outstanding [17], this document uses the fine-grained parallel model (*Parallel Genetic Algorithm based Grid*) to implement a QAP solution.

2.2.1 Parallel Genetic Algorithm based Grid (PGAG)

Individuals are placed in a two-dimensional grid, one in particular per cell. The evaluation of individual fitness is done simultaneously and selection and crossover are carried out within each neighborhood. This model allows maintaining the diversity in the population by controlling the convergence of the premature individual and allows exploring different parts of the search space.

Four different neighborhoods are usually considered by PGAG, [8]. The Grids are toroidal. (Figure 1 shows of these topologies):

- $4n$: Horizontal and vertical neighbors.
- $8n$: All the neighbors of distance 1.
- $16n$: In all 8 directions, neighbors with distance 2.
- $24n$: In all 8 directions, neighbors with distance 2 plus all the neighbors of distance 1.

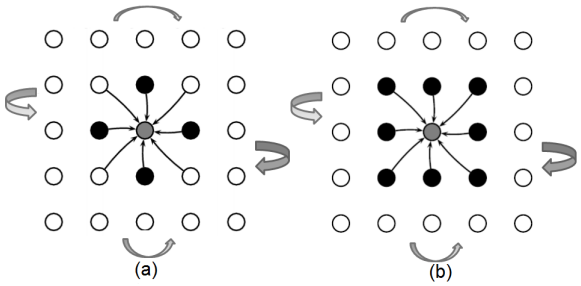

Figure 1: Cell Topologies. (a) $4n$, (b) $8n$. From [7]

2.3 Graphics Processing Unit (GPU)

Parallel multiprocessing architectures like Graphic Processing Units (GPU) [4] have significantly evolved in the last nine 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 (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.

2.4 Local Heuristics Search 2-opt

The heuristic consists of performing all pairwise exchanges of all possible facilities on each of the locations in a particular permutation (individual).
The solution is updated by a better as long as $\Delta_{ij} < 0$ in the following inner product formula (i, j are exchanges installations).

$$\Delta_{ij} = \Big((F_{i\cdot} - F_{j\cdot}) + (F_{\cdot i} - F_{\cdot j})\Big) \bullet \Big((XD)^t_{\sigma(j)} - (XD)^t_{\sigma(i)}\Big)$$

with $f_{ij} = f_{ji} = 0$.
$A_{k\cdot}$ indicates the row k of the matrix A and $A_{\cdot k}$ indicates the columnm k of the matrix A. In the formula the distance matrix D must necessarily be a matrix symmetric.

3 Implementation

The implementation was achieved using the grid model (fine-grained parallel model), improved from the use of the Mutation and Transposition Genetic Operators and the Local Heuristics Search 2-opt to solve eight different instances of QAP corresponding to prominent benchmark problems found into QAPlib [1], they are:

- Chr25a (One matrix is the adjacency matrix of a weighted tree the other that of a complete graph [3], graph Size: 25)
- Els19 (The data describe the distances of 19 different facilities of a hospital and the flow of patients between those locations [5], graph Size: 19)
- Esc32a (These examples stem from an application in computer science, from the testing of self-testable sequential circuits [6], graph Size: 32)
- Had20 (The first matrix represents Manhattan distances of a connected cellular complex in the plane while the entries in the flow matrix are drawn uniformly from the interval $[1, n]$ [8], graph Size: 20)
- Kra32 (The instances contain real world data and were used to plan the Klinikum Regensburg in Germany [11], graph Size: 32)
- Nug30 (The distance matrix contains Manhattan distances of rectangular grids [12], graph Size: 30)
- Rou20 (The entries of the matrices are chosen from the interval $[1, 100]$ [14], graph Size: 20)

- Scr20 (The distances of these problems are rectangular [16], graph Size: 20)

For each problem instance a two-dimensional GPU grid of size 10×10 was used, where each GPU blocks consists of n GPU threads (n is the size of the QAP). Each GPU block corresponds to an individual population and each GPU thread corresponds to a gene on the chromosome that represents each individual. An integer coding was used. If A is the chain (chromosome) that represents this coding, then, the position i of the A chain corresponds to i installation and the $A[i]$ content of the chain (gene $A[i]$) corresponds to the $\sigma(i)$ location. Figure 2 shows such a case.

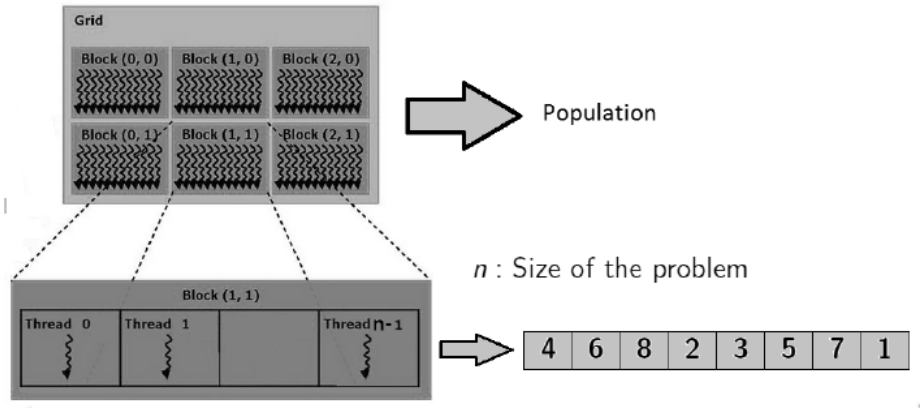

Figure 2: Population

The evaluation of the individual aptitude of the population was made with the *trace* alternative formula for the QAP. This formula makes better use of the multiprocessing features of the GPU.
For evaluation of individual fitness, the same two-dimensional GPU grid was taken, but each GPU block consists of a $(n \times n)$-sized two-dimensional array of GPU threads. Each two-dimensional GPU thread is either a component of the permutation matrix X, a component of the Flow matrix F, or a component of the Distance matrix D. Figure 3.
The Flow and Distance matrices reside in the constant memory space of the GPU, to speed up the calculations.
The genetic operators considered in the PGA are the basic selection, crossover, mutation and transposition operators. All operators are implemented at block level in the GPU similar to the configuration presented in Figure 2.
The selection is Tournament. This method compares individuals of the population (confronting them by the fitness) and choose the fittest, the comparison is made by couples of individuals (a binary tournament).

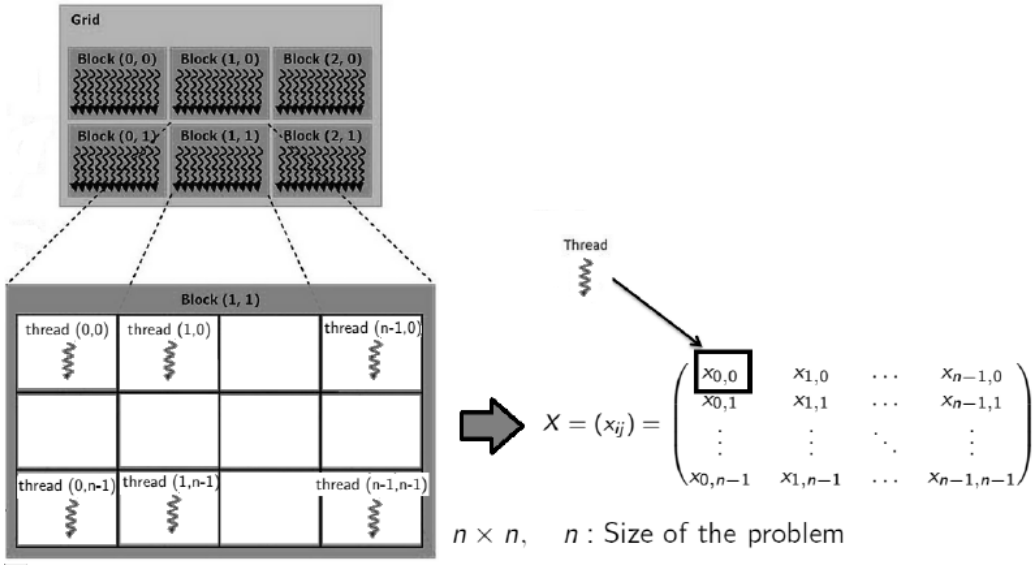

Figure 3: Fitness Evaluation on GPU

The crossover is a Modified order crossover (MOX). A common crossing point is selected for both parents, the genes of the first parent are stored to the left of the crossing point and then the remaining genes of the second parent (this second parent is chosen according to a neighborhood $8n$ topology to the PGAG) are copied in the order they are to obtain offspring. The MOX on GPU is done as in Figure 4.

The mutation is an Exchange mutation (EM). This type of mutation selects two genes (GPU threads) and exchanges them, Figure 5.

Transposition simply reverses the chromosome genes (GPU threads) between two points randomly generated, Figure 6.

After each iteration, the best individual in the population always survives, it is considered an elitist Genetic Algorithm model.

The crossover probability rate is 0.6 in all GPU blocks (each individual in the population) and the mutation probability rate is 0.1 in all blocks except in five of them where it is 0.6; it is important to note that in conjunction with the elitist model used, it is productive to consider a high mutation probability (0.6) to gain diversity [7]. The rate used in the transposition operator is 0.5 in all GPU blocks.

Finally, the local heuristics search 2-opt is implemented in order to exploit in greater depth promising regions already explored by the genetic algorithm. It is easy to evaluate the increase or decrease in the fitness function produced by these exchanges in the block-level GPU (i.e. all possible exchanges on each of the individuals in the GPU grid considering a configuration as in Figure

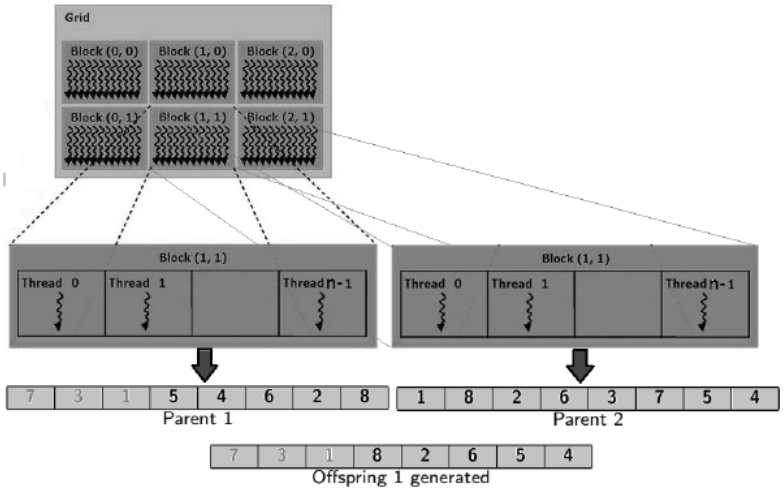

Figure 4: MOX on GPU

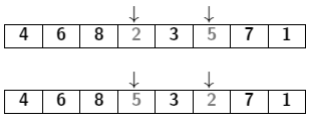

Figure 5: EM

4) through the inner product just like previously mentioned. The procedure implemented is described in algorithm 1.

Algorithm 1. Improved Grid Model

For each cell i in the grid **DO** in parallel
    Assign a random individual
End **DO** in parallel
WHILE termination condition not met
    For each cell i **DO** in parallel
        Evaluate individual i
        Select a neighboring individual k from i
        Cross individuals i with k
        IF the offspring is better than i
            Assign the best offspring to i
        End **IF**
        Mutate i with fixed probability
        Transpose i with fixed probability
        Apply 2-opt to i
    End **DO** in parallel

End **WHILE**.

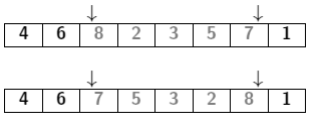

Figure 6: Transposed

4 Results and conclusions

In Table 1 we can see the best solution for each of the eight chosen benchmark problem reported into QAPlib; the average solution found for our parallel model including the quantity of iterations (*#itr*), the *time* elapsed to reach such response (time in seconds) and the *gap* between the found solution and the best reported solution, according to the formula:

$$gap = \frac{\text{Average solution found} - \text{Best solution reported into QAPlib}}{\text{Best solution reported into QAPlib}}$$

The average solution reported by our algorithm is the average solution found in five executions for each problem. *#itr* correspond to the smallest number of iterations and *time* therefore the shortest time elapsed. The algorithm was executed in an Intel®Core™ i7 - 4700HQ CPU @ 2.40GHz, RAM 8 GB y GPU Nvidia GeForce GTX 760M.

Name instance	**Size of instance**	**Best known solution**	**Solution found**	***#itr.***	***time* (seconds)**	***GAP***
Chr25a	25	3796	3847	119	33.34	0.013
Els19	19	17212548	17212548	3	1.17	0
Esc32a	32	130	130	22	12.49	0
Had20	20	6922	6922	2	0.49	0
Kra32	32	88700	88700	10	5.75	0
Nug30	30	6124	6124	16	7.42	0
Rou20	20	725522	726610	7	1.51	0.001
Scr20	20	110030	110030	6	1.48	0

Table 1: Obtained results

Name instance	**Best permutation obtained in QAP**
Chr25a	24 11 4 2 17 3 15 7 19 9 13 5 14 22 23 18 12 0 20 10 16 1 21 8 6
Els19	8 9 6 18 13 17 12 16 5 10 3 4 11 7 15 14 0 1 2
Esc32a	14 6 22 20 4 12 3 27 5 16 13 10 15 7 23 26 9 21 29 28 19 8 18 17 24 11 25 2 31 1 0 30
Had20	7 14 15 13 18 5 6 16 0 11 9 10 4 19 1 2 3 8 17 12
Kra32	11 5 8 1 9 4 22 26 27 18 7 3 23 15 0 6 10 14 16 25 21 2 17 24 29 13 30 19 20 12 28 31
Nug30	19 13 2 26 17 14 3 29 15 10 21 22 24 18 6 7 0 16 27 28 8 9 23 25 20 1 12 5 11 4
Rou20	0 18 1 13 9 15 10 19 8 4 6 3 7 17 14 2 11 16 12 5
Scr20	16 5 8 6 0 4 1 2 14 9 18 11 17 13 12 19 15 7 10 3

Table 2: Best Tour obtained in QAP

It can be concluded that using a coarse-grained parallel model or a hybrid between both, as Grisland did [7] it is possible to tackle problems greater than those considered, also, the use of local heuristics search k-opt ($k > 2$) it will surely help find good results for these greater instances of the problem.

References

[1] M. Anjos, Qaplib home page. http://anjos.mgi.polymtl.ca/qaplib/

[2] R. E. Burkard, Quadratic Assignment Problem, Chapter in *Handbook of Combinatorial Optimization*, P. M. Pardalos, D-Z. Du, R. L. Graham, ed., Springer, New York, 2013, 2741-2814. https://doi.org/10.1007/978-1-4419-7997-1_22

[3] N. Christofides and E. Benavent, An exact algorithm for the quadratic assignment problem on a tree, *Operations Research*, **37** (1989), no. 5, 760-768. https://doi.org/10.1287/opre.37.5.760

[4] N. Cuda, CUDA GPUs, 2016. https://developer.nvidia.com/cuda-gpus

[5] A. N. Elshafei, Hospital layout as a quadratic assignment problem, *Operations Research Quarterly*, **28** (1977), 167-179. https://doi.org/10.2307/3008789

[6] B. Eschermann and H.-J. Wunderlich, Optimized synthesis of self-testable finite state machines, *[1990] Digest of Papers. Fault-Tolerant Computing:*

20th International Symposium, (1990), 390-397. https://doi.org/10.1109/ftcs.1990.89393

[7] J. Gomez, R. Poveda and E. Leon, Grisland: A parallel genetic algorithm for finding near optimal solutions to the traveling salesman problem, *Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference-GECCO '09*, (2009), 2035-2040. https://doi.org/10.1145/1570256.1570272

[8] S. W. Hadley, F. Rendl and H. Wolkowicz, A new lower bound via projection for the quadratic assignment problem, *Mathematics of Operations Research*, **17** (1992), 727-739. https://doi.org/10.1287/moor.17.3.727

[9] K. A. De Jong, W. M. Spears and D. F. Gordon, Using genetic algorithms for concept learning, *Machine Learning*, **13** (1993), no. 2-3, 161-188. https://doi.org/10.1007/bf00993042

[10] T. C. Koopmans and M. Beckman, Assignment problems and the location of economic activities, *Econometrica*, **25** (1957), 53-76. https://doi.org/10.2307/1907742

[11] J. Krarup and P. M. Pruzan, Computer-aided layout design, *Mathematical Programming in Use*, Mathematical Programming Studies, Vol. 9, Springer, Berlin, Heidelberg, 1978, 75-94. https://doi.org/10.1007/bfb0120827

[12] C. E. Neugent, T. E. Vollman and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, *Operations Research*, **16** (1968), 150-173. https://doi.org/10.1287/opre.16.1.150

[13] P. M. Pardalos and J. Xue, The maximum clique problem., *Journal of Global Optim.*, **4** (1994), 301-328. https://doi.org/10.1007/bf01098364

[14] C. Roucairol, *Du Sequentiel au Parallele: La Recherche Arborescente et Son Application a la Programmation Quadratique en Variables 0 et 1*, Thse d'Etat, Universit Pierre et Marie Curie, Paris, France, 1987.

[15] S. Sahni and T. Gonzalez, P-complete approximation problems, *Journal of the ACM*, **23** (1976), 555-565. https://doi.org/10.1145/321958.321975

[16] M. Scriabin and R. C. Vergin, Comparison of computer algorithms and visual based methods for plant layout, *Management Science*, **22** (1975), 172-187. https://doi.org/10.1287/mnsc.22.2.172

[17] M. Tomassini, A survey of genetic algorithms, Chapter in *Annual Reviews of Computational Physics III*, World Scientific, 1995, 87-118. https://doi.org/10.1142/9789812830647_0003

Received: November 14, 2017; Published: November 27, 2017