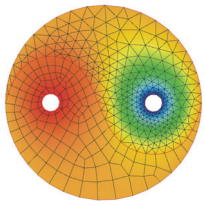
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 11, 2011, No. 3



EASEA: A GENERIC OPTIMIZATION TOOL FOR GPU MACHINES IN ASYNCHRONOUS ISLAND MODEL

LAURENT A. BAUMES*[1], FREDERIC KRUGER[2], PIERRE COLLET[2]

[1] ITQ, UPV-CSIC, Valencia, Spain
[2] Univ. Strasbourg, LSIIT, FDBT, Illkrirch, France
**Corresponding author: baumesl@itq.upv.es*

Abstract

Very recently, we presented an efficient implementation of Evolutionary Algorithms (EAs) using Graphics Processing Units (GPU) for solving microporous crystal structures. Because of both the inherent complexity of zeolitic materials and the constant pressure to accelerate R&D solutions, an asynchronous island model running on clusters of machines equipped with GPU cards, i.e. the current trend for super-computers and cloud computing, is presented. This last improvement of the EASEA platform allows an effortless exploitation of hierarchical massively parallel systems. It is demonstrated that supra-linear speedup over one machine and linear speedup considering clusters of different sizes are obtained. Such an island implementation over several potentially heterogeneous machines opens new horizon for various domains of application where computation time for optimization remains the principal bottleneck.

Key words: GPU, Evolutionary Algorithms, Island Model, Parallelism, Zeolite Materials

1. INTRODUCTION

Cloud computing is now coming and there is still a lack of generic optimization software allowing to efficiently and effortlessly run over such infrastructure. Therefore, this paper shows how machines connected through a local network or Internet may be employed adequately using EASEA[1] (EAsy Specification of Evolutionary Algorithms) massively parallel platform (Collet et al., 2000). On the other hand, zeolite discovery is a very tedious domain of investigation which may last several years. High Throughput (HT) techniques (Farrusseng et al., 2002; Farrussen et al., 2003; Ausfelder et al., 2011a; Ausfelder et al., 2011b; Jiang et al., 2011) allow reducing the experimental effort during the synthesis and characterization steps. New adapted methodologies able to correctly support the flow of data have been provided for both synthesis optimization (Corma et al., 2006; Baumes et al., 2007; Serra et al., 2007; Baumes et al., 2009; Baumes & Collet, 2009) and crystallographic phases identification (Baumes et al., 2008; Baumes et al., 2009; Baumes et al., 2011a, Baumes et al., 2011b) in order not to slow down the whole process. However, two different steps remain critical: (*i*) The first one, before starting any experiments, is the choice and design of one given Structure Directing Agent (SDA) or so-called "template" which will direct the synthesis toward one framework or another; (*ii*) the second, at the very end of the process when a new material has been found, consists in solving its crystal structure, see figure 1. Here, the new functionality of EASEA is expected to accelerate the latter by several orders of magnitude using the same methodology presented earlier (Baumes et al., 2007; Maitre et al., 2009a;

[1] http://sourcefore.net/projects/easea

ISSN 1641-8581

Maitre et al., 2009b; Krüger et al., 2010; Baumes et al., 2011b; Jiang et al., 2011) while removing the substantial difficulties faced by non expert programmers willing to run parallel EAs.

parameters for a specified evolutionary engine to evolve a population of individuals. In literature (Baumes et al., 2007; Maitre et al., 2009a; Maitre et al., 2009b; Krüger et al., 2010; Baumes et al.,

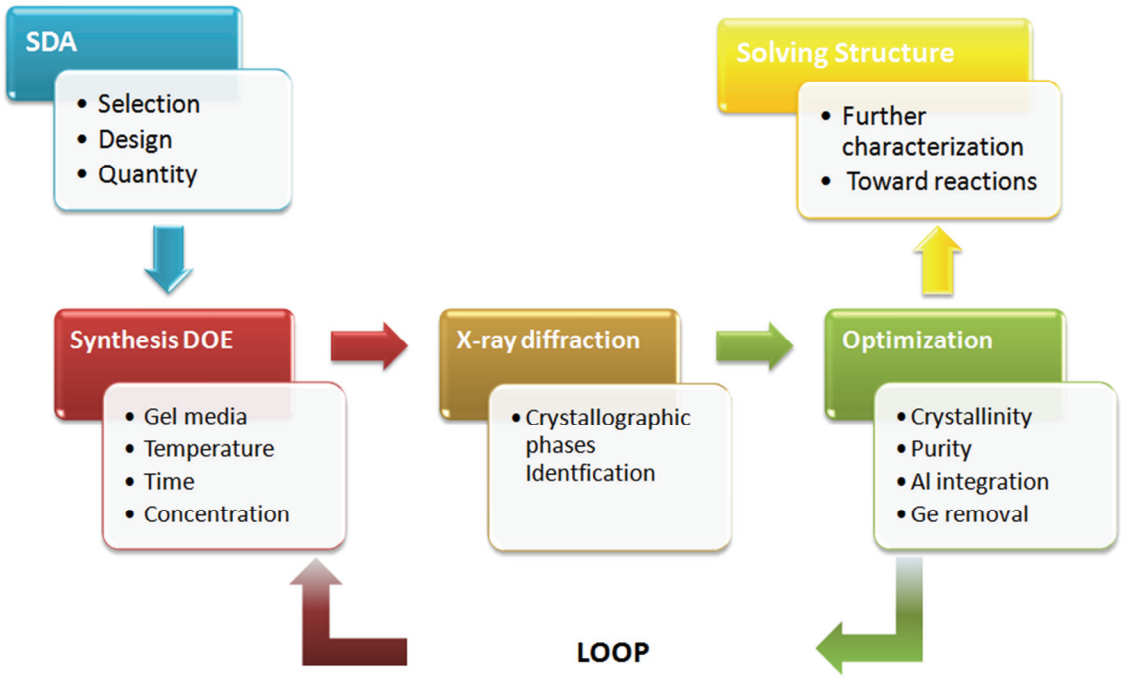


Fig. 1. *Flowchart representing the different steps required for zeolite discovery.*

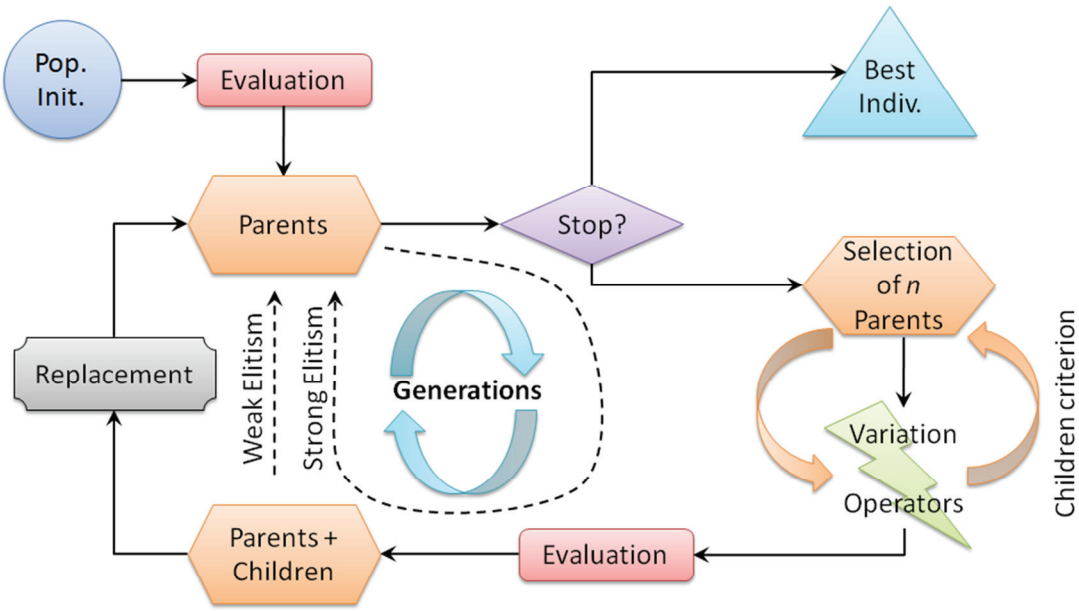


Fig. 2. *Flowchart of an evolutionary algorithm implemented by EASEA.*

The EASEA platform allows a quick and simple EA formalism through supplying problem-specific pieces of code only, i.e. individual representation, initialization, mutation, recombination, and fitness evaluation, in an `.ez` file that also contains default

2011b), an implementation of an EA using GPU is presented. The efficiency of the technique is due to the transfer of the most time consuming part of the algorithm on the GPU, e.g. the individual fitness evaluation, while all the rest is maintained on the CPU, see figure 2.

In materials science, knowledge of the structure at an atomistic/molecular level is required for any advanced understanding of its performance. It is therefore essential that methods to study structures are developed due to the intrinsic link between the structure of the material and its useful properties. Despite the fact that various methods are existing, the main problem remains the tremendous increase of computational time when the number of atoms to be considered is relatively high. Therefore, one solution is to be able to run these techniques on a massively parallel architecture if the algorithm allows it. EAs are intrinsically parallel methods and consequently, EASEA has been designed in order to best take advantage of this characteristic at all the different levels: SPMD, SIMD, and MIMD as demonstrated later. The methodology proposed elsewhere allows generating various possible crystalline solutions by optimizing the geometry of the zeolites under the constraints defined by the sample X-ray diffraction (XRD) indexing, *i.e.* lattice parameters (unit cell dimensions and angles, space group), and from density data (number of T atoms per unit cell). Among the different stable structures generated, the framework corresponding to the new materials is found through examination and comparison of theoretical and experimental XRD, see figure 3. The extension of this procedure which directly integrates theoretical and experimental comparison of XRDs as part of the fitness function is not presented here.

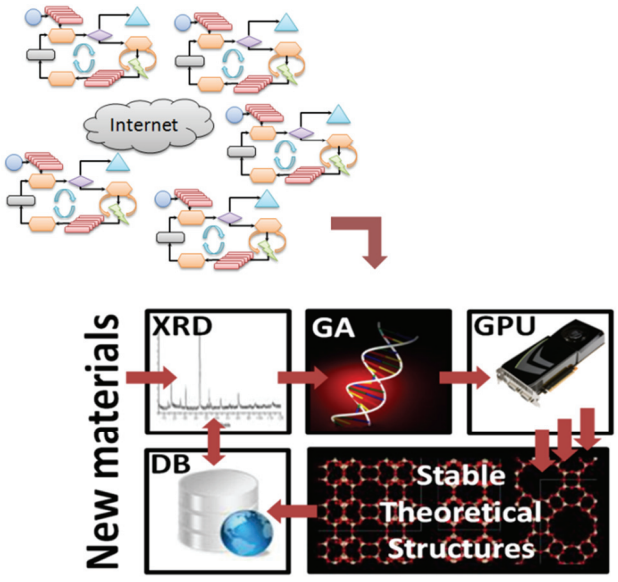


Fig. 3. *Bottom - Zeolite structure generation using an evolution algorithm and GPU hardware. Top – Use of Island Model to accelerate GA search.*

2. EASEA

Code for many different paradigms (Genetic Algorithms, Evolution Strategies, CMA-ES, memetic algorithms, tree-based genetic programming, linear genetic programming, and in a short future, differential evolution) can be produced either for a standard CPU, e.g. sequential execution on one core, or over one or several NVidia GPU cards depending on the `-cuda` option invocation on the command line. As mentioned earlier, EASEA parallelization over GPU cores has already been extensively described in a number of recent papers (Baumes et al., 2007; Maitre et al., 2009a; Maitre et al., 2009b; Krüger et al., 2010; Baumes et al., 2011b; Jiang et al., 2011), and consequently will focus on the parallelism established at higher levels, *i.e.* clusters and group of clusters.

2.1. Island Model and related works

Tanese et al. are two of the earliest parallel implementations (Pettey et al., 1987; Tanese, 1987). The population of a GA was split into a relatively small number of sub-populations and each processing element in the architecture was assigned an entire sub-population and executed a standard GA. In 1992 and then two years later, Gordon (Gordon et al., 1992) and Adamidis (1994) consecrated the term of island model parallel GA. Island model genetic algorithms are based on independent GAs which evolve separately, and intermittently exchange genetic material, see general scheme in figure 4. However, individuals should not be exchanged among islands too often: local exploitation should take place in order so that a new spot is well explored before the island migrates to another place towards the direction of another good immigrant indicating that infrequent communication is actually an advantage over frequent communication. As communication is usually what prevents parallel machines from yielding a linear speedup with the number of machines, EASEA migration scheme can take place asynchronously and in a loosely coupled manner. If the main process stops and waits for all evaluations to complete before moving to next generation, the algorithm is said to be synchronous, asynchronous otherwise. Theoretical studies on the number and the size of populations have been done (Whitley et al., 1999). Other studies show the influence of different island model parameters (Cantú-Paz, 2000). Alba and Troya studied asynchronism (Alba & Troya

1999), Branke *et al.* studied the influence of heterogeneous networks on island models (Branke et al., 2004). Some research has been done on parallelizing evolutionary computation using an island model on a GPGPU card, but this research was limited to a single machine equipped with a GPGPU card that hosted all the islands (Luong et al., 2010).

and therefore robust. As a matter of fact, an island can interrupt its search and then start again later on without disturbing the other islands. In addition, it will receive good immigrants from other machines allowing it to catch up with the other islands and continue the search where it might have left off to find good spots again. This statement also means

```
Islands Model(E, N, μ)
{
  Concurrently for i ← 1 to N
    Initialize(P_i, μ);                          Initialization

  for epoch ← 1 to E
  {
    Concurrently for i ← 1 to N
      Sequential_EA(P_i, G_i);                   Isolated Evolution for an epoch

    Concurrently for i ← 1 to N
      P_i' = P_i;
    Concurrently for i, j ← 1 to N
      P_i' ∪= Migration(P_i, S_i,j);             Migration
    Concurrently for i ← 1 to N
      P_i = Assimilate(P_i');
  }

  ExtractProblemSolution;                        Solution Extraction
}
```

Fig. 4. *Island Model general scheme with P the population, P_i are individual sub-populations, M=N x μ where M is the overall size of population, N is the number of islands, and μ is the size of sub-population. Epoch is the period of isolated evolution and G_i is a number of generations each island evolves in isolation. S is a NxN matrix (0 diagonal) and $S_{i,k}$ are the number of individuals from P_i to migrate into P_k at the end of each epoch.*

2.2. Our implementation

Models may be expressed as nested higher-order functions and realised as the corresponding nested algorithmic skeletons. The default island model implemented by the EASEA language is quite basic. It allows to periodically send (and receive) an individual to (from) one of several IP addresses listed in a file. The configuration of the EASEA island model relies on two main parameters, the name of the file containing the IP addresses and the migration probability. Several islands usually run on the same machine according to traditional approaches. But the EASEA island model only lets a single island to run per machine and relies on the establishment of connexions with other machines to form a multi-island system. Individuals are exchanged between the islands using a connexion-less UDP/IP protocol. No connexion implies that the process is asynchronous

that new islands can join the search at any moment without disturbing the overall process while no time is lost in island synchronization. Another point of interest is that EASEA produces code for any OS, and consequently heterogeneity of OS is supported *via* the internet over the entire world enabling very effective problem solving strategies. Algorithms may be different depending on machines they run on: slower machines can have an algorithm doing exploration while faster ones can concentrate on exploiting good spots. Such a configuration is actually tested for the most difficult zeolites stored in IZA database before a new one is synthesized in the Instituto de Tecnologia Quimica in Valencia, Spain, see figure 5. For this, the IP address can be repeated in the IP file allowing an implementation of weighted edges connecting machines. As communication is not necessarily symmetric, this allows implementing a unidirectional flow of individuals between slow exploratory machines who's job it would

be to find good spots and fast exploitation machines that could concentrate on finding the local minimum. However, the fast machines may not send back individuals to the slow machines so as to prevent premature convergence in the cluster of slow machines and spoil the exploration.

on an island starts, a thread is launched that is in charge of the reception of the incoming immigrants. When a new individual arrives, a selector decides who in the population will be replaced. The algorithm will check every generation if new individuals arrived and start the integration process for each on

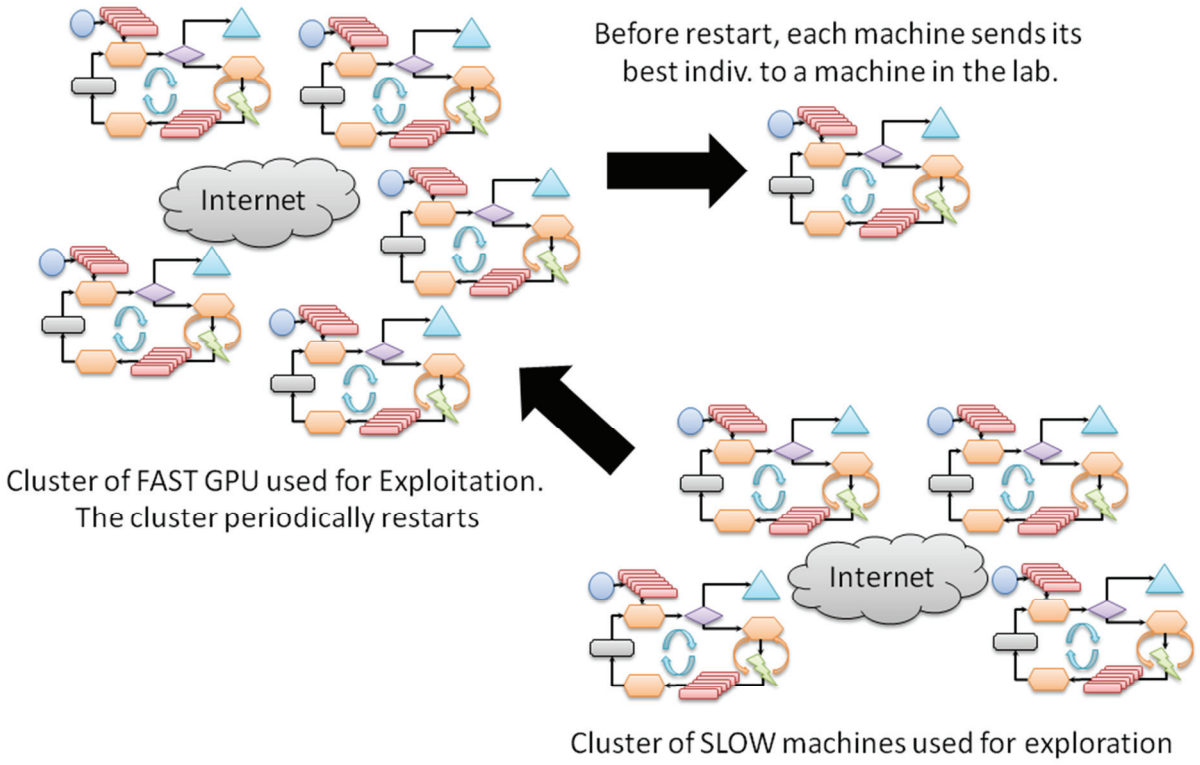


Fig. 5. *Heterogeneity of machine and Island configuration for efficient optimization.*

There are known effects of magnitude and frequency of migration. For example, limited migration between populations capitalizes on 'punctuated equilibria' effect (Eldredge & Gloud, 1972). More migrants or shorter epochs have the effect of precluding isolated evolution on separate islands. Genetic diversity vanishes quickly and the behaviour approximates that of a classic EA running on the whole population. On the other hand, insufficient migration keeps the islands too far apart. The genetic richness of the neighbouring populations doesn't have enough chance to spread out. In this regime the parallel run simulates N independent runs with population size N times smaller. In EASEA, the migration function is called at every generation with a probability p set by the user. A destination island is chosen randomly amongst the list of clients. Once a destination is chosen, n individuals are selected amongst the population using different selection operators. The selected individuals are then serialized and sent to the destination island. When a run

of them. Selection may be defined by many different manners. For example, one could send a migrant to that one island whose genetic material is most different (Lin et al., 1994) or to that island that has maintained the most diversity (Munetomo et al., 1993).

The implementation of the EASEA island model described above is simple, robust and versatile. Complex topologies can easily be designed using the EASEA island model. For instance, it would be simple to create a ring shaped island model by just giving each island the address of the following island. Communication rate can be increased or decreased by changing the migration probability: for instance, the centre island in a star shaped topology may require more frequent communications with the other islands. One could also imagine to modify the migration rate based on some more elaborate strategy (increase or decrease migration with the number of generations).

3. EXPERIMENTS

In this section, a benchmark and a real-world problem are tested using the EASEA GPU island model on a cluster of 20 PCs each containing an NVIDIA GTX275 GPU card, for an advertised computing power of around 20 TFlops.

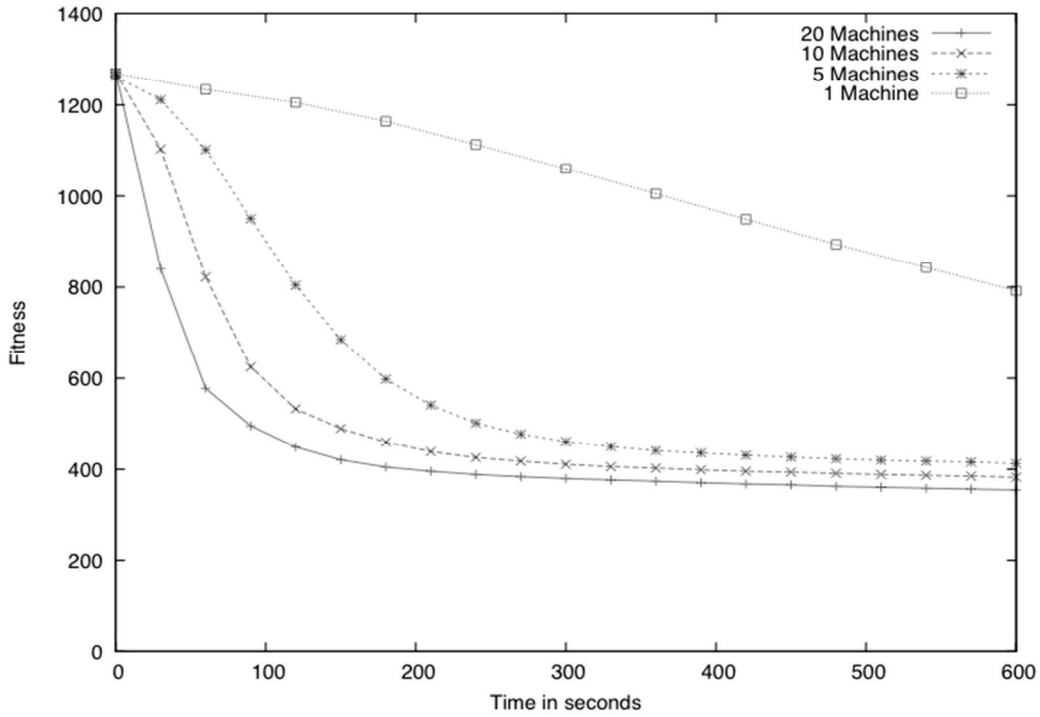


Fig. 6. *Evolution of the best fitness on different cluster sizes. The results represent an average of 20 runs.*

3.1. Obtained speedup on the Weierstrass Benchmark

As a benchmark function to put the EASEA island model to the test, we used a Weierstrass-Mandelbrot function, whose irregularity can be adjusted thanks to its Hölder coefficient h. A value of 0.5 for Hölder coefficient is usually used. But after several tests it appeared that the function was too simple, so irregularity was increased by using a 0.35 Hölder coefficient and 120 iterations along with 1000 dimensions.

3.2. Observed speedup on the 20 machines cluster

In order to study the efficiency of the EASEA island model, the Weierstrass-Mandelbrot benchmark function was tested on different cluster sizes, each machine of the cluster hosting a single island. To be comparable, all experiments are done at a constant population size, meaning that each island of a 20-machine experiment will contain 20 times less individuals than the same experiment on a single machine, i.e. 1 machine used 81920 individuals, 16394, 8192, and 4096 for 5, 10, and 20 respectively. The topology used for the experiments is a fully connected network and all presented results were obtained over an average of 20 runs (no machine can send an individual to itself). Figure 6 shows the evolution of the average of the best fitness over 10 minutes for the Weierstrass-Mandelbrot benchmark. As expected, the more machines in the cluster, the faster a better solutions is obtained.

In order to study the efficiency of the EASEA island model, the Weierstrass-Mandelbrot benchmark function was tested on different cluster sizes, each machine of the cluster hosting a single island. To be comparable, all experiments are done at a constant population size, meaning that each island of a 20-machine experiment will contain 20 times less individuals than the same experiment on a single machine, i.e. 1 machine used 81920 individuals, 16394, 8192, and 4096 for 5, 10, and 20 respectively. The topology used for the experiments is a fully connected network and all presented results were obtained over an average of 20 runs (no machine can send an individual to itself). Figure 6 shows the evolution of the average of the best fitness over 10 minutes for the Weierstrass-Mandelbrot benchmark. As expected, the more machines in the cluster, the faster a better solutions is obtained.

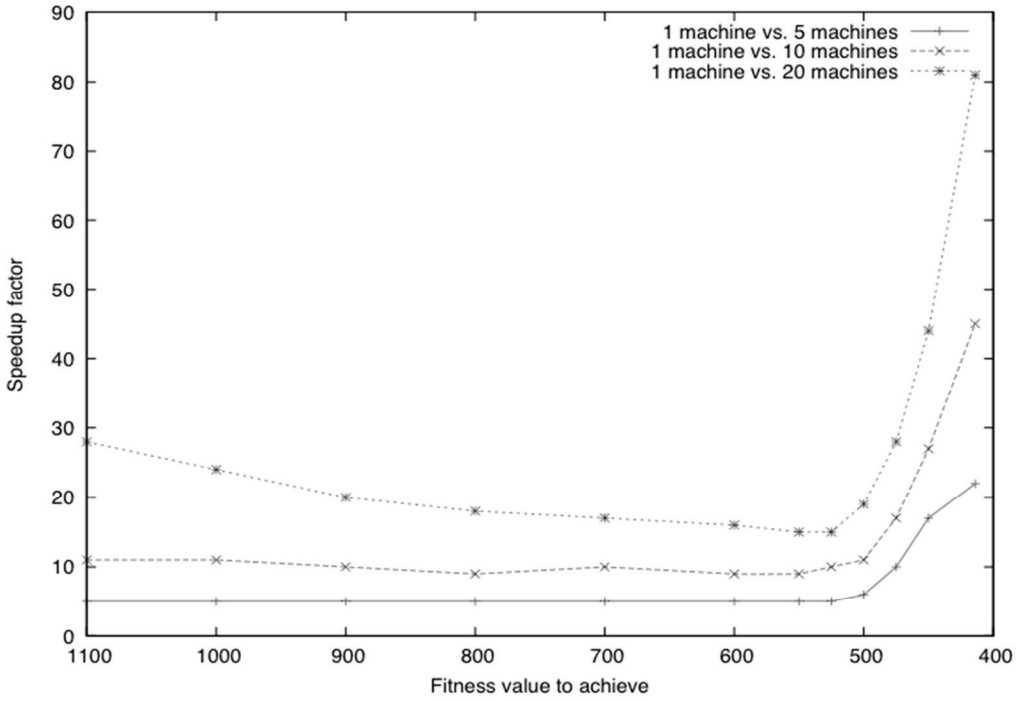


Fig. 7. *Speedup factor achieved comparing 1 machine vs. 5, 10 and 20 machines clusters. These results represent an average of 20 runs.*

It is possible to visualize the obtained speedup by looking at how long it takes each cluster configuration to reach predefined values such as 1100, 1000, 900… It was not possible to go down to value 400 as some single machines did not improve their results after more than 10 hours. The minimum value found on one of the 20 runs for a single machine was 414. Figure 7 can be read the following way: it was 5 times faster for a 5 machines cluster to obtain a fitness of 1100, compared to one machine only, meaning that a linear speedup is obtained for 5 machines. Indeed, the $\times 5$ speedup is quite constant until value 500 after which it increases up to more than 20 for value 414 (these are all average values over 20 runs).

The speedup factor for a 10 machines cluster is quite similar: around $\times 10$ speedup from value 1100 down to value 500 after which speedup rises up to around $\times 45$. With 20 machines, speedup starts with a supra-linear value of 28, before going down to slightly under 20 and rising up again at value 500, up to slightly above $\times 80$. What can be observed here is that speedup is more or less linear with the number of machines until value 500, after which single machines with an island of 81920 individuals stagnate. This is the point where multi-island models show their advantage as they help each other out of local optima. Therefore the speedup of multi-island models over a single island gets very supra linear.

It is important to remember that the curves presented above are for GPU-islands, these GPU-islands being already x160 faster than a sequential execution of the same code on a CPU (figure not shown). Therefore, a single one-day run on this cluster for this problem would be equivalent to no less than 35 years of computation on a single island without GPU. Though it would probably be much more as value 414 was obtained by the 20 machines cluster in less than 3 minutes and the speedup slope is quite steep.

4. USING THIS SETUP TO LOOK FOR A ZEOLITE STRUCTURE

Even though benchmark problems are perfect to test performances, real-world problems often don't behave the same way. A good challenge was therefore to try the island model on a chemistry problem: finding the structure of a new zeolite.

Zeolites are crystalline materials with regular structures consisting of molecular-sized pores and channels. These crystals are widely spread and used in a lot of important industrial applications such as in the field of absorption, ion exchange, and heterogeneous catalysis, as well as in health, sensors, and solar energy conversion. They are made of tetrahedrons of oxygen atoms that contain a single silicon or aluminium atom in their centre. The problem posed by prediction is to obtain approximate models

in a reasonable time, which can then easily be subsequently refined by routine modelling techniques. The proposed structure prediction approach is expected to provide stable candidate solutions without comparing their theoretical diffraction data to the experimental one during the evaluation of goodness. Next, the different potential solutions proposed by the algorithm are refined by a routine based on interatomic potential technique using the GULP code, and finally, their theoretical x-ray diffraction (XRD) compared to the experimental one. The entire methodology is explained in previous papers (Baumes et al., 2007; Maitre et al., 2009a; Maitre et al., 2009b; Krüger et al., 2010; Baumes et al., 2011b). Here, we test the island model using the MFI zeolite which contains 96T in the unit cell. Note that in the version used for this test, after all atoms have been randomly placed in the asymmetric unit during the initialization procedure, each atom is iteratively and randomly selected and its position is translated toward the closest atom already present which is not tetra coordinated. The final position is the optimal distance defined at 3.07A. The procedure stands for an improved initialization made once at generation 0. Also, Wickoff positions have also been integrated, allowing atoms being at a distance inferior to 0.8A to be automatically placed at the corresponding special position onto the symmetry element defined by the space group before the assessment of the solution. Note that the new position is virtual, *i.e.* calculated, in order to assess the structure viability as `{x,y,z}` coordinates are not modified in the genome code. The fitness function computes an approximated value of the real energy in the system. The function is based on geometric terms. The goal of the evolutionary algorithm is to minimize this fitness.

A simple fully connected island model was not efficient enough to find interesting results for this very complex real world problem. Two different clusters were created out of the 20 machines: one cluster of 16 machines that would do some exploration and one cluster of 4 machines with an algorithm fit enough to exploit the best spots. The machines of the first cluster would periodically send their best individuals to the 4 other machines while the 4 machines would exclusively exchange individuals between themselves. Their main goal would be to find the best local value around the individuals sent by the cluster of 16 machines. If after a while they didn't manage to find any improvements, the 4 machines would simply restart simultaneously and wait for new suggestions coming from the 16 other machines. Periodically, the 4 machines would send the best individuals they found to an independent slow machine in our lab that was not part of the optimization process. It would simply run to serve as an archive and collect the best individuals found to date. The same settings were used as for the Weierstrass test bench (population sizes depending on the number of machines in each tested cluster configuration).

4.1. Results

Figures 8 and 9 show the evolution of the best fitness and the respective speedup for the crystal structure prediction problem for runs on a cluster of 1, 5, 10 and 20 machines. The 20 machines cluster (and its special topology) outperformed all the others, giving better results for the same computation time. The correct structure which has a fitness value equals to zero is found as the same framework, e.g. connectivity, may be found around 2 due to flexibility and deformation of bonds.

It is interesting to note that where the speedup for 5, 10 and 20 machines is linear for a target value of 5, it decreases after this value. Indeed, apart for 5 machines, beyond a value of 5, the speedup for 10 machines is just about ten, and the speedup for 20 machines stays around 15. This means that down to value 5, the problem is very irregular (as for the Weierstrass function), so the island model is very efficient. Beyond, it seems that the problem is less irregular, so it is not 20 times more efficient to have 20 machines work in parallel. However, if 20 machines are available, they will nevertheless go faster than 5 only.

So, even though the irregularity of the problem seems to match well a 5 (or 10) machines configuration and less a 20 machines configuration, 20 machines is better to have than 10 only. It is only slightly less efficient.

Finally, on this problem, there is no stage at which single machines stagnate on local optima. Therefore, the kind of supra-linear speedup that was observed on the Weierstrass-Mandelbrot function is not to be found here (or maybe more time should be given to see the phenomenon appear).

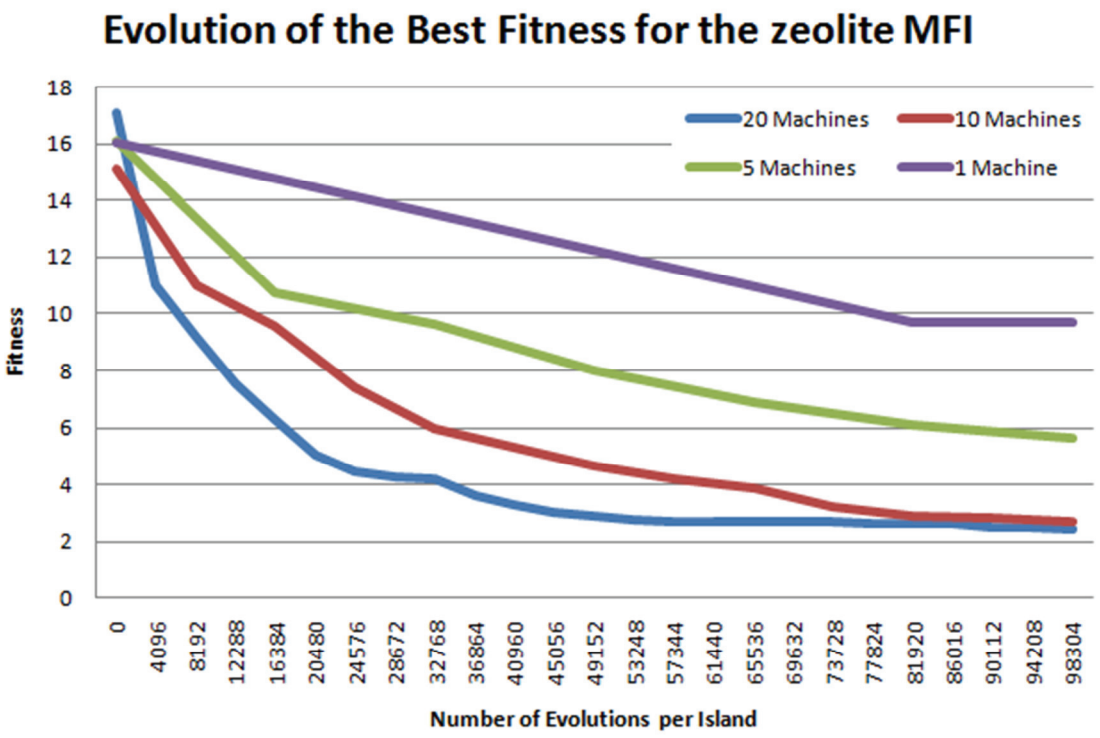


Fig. 8. *Evolution of the best fitness for the real world problem.*

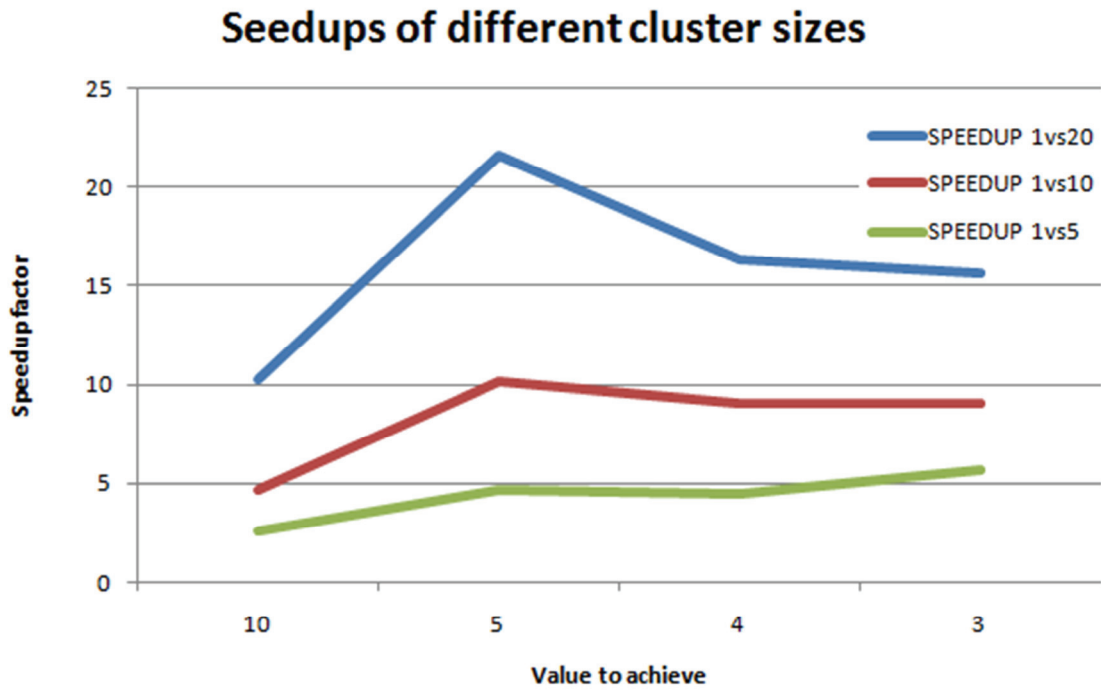


Fig. 9. *Observed speedup for the real world problem.*

5. CONCLUSION AND EXTENSION

To summarize, the idea behind Island model is to split the population in many sub-populations – islands, alternate periods of extensive isolated evolution with migration, during isolated evolution each process runs on its island a full blown EA, and at certain times, few individuals migrate between islands. The advantage of such an approach is that it is uniquely suited for message passing parallel systems and more then hardware accelerators for EAs. On the other hand, the price to be paid is defined by more complex design decisions due to increased number of parameters as well as dynamics of multiple EAs running in parallel. EASEA allows automat-

ically parallelizing the evaluation stage of evolutionary algorithms using such scheme, which nevertheless needs to be controlled by the user. EASEA platform now integrates a basic yet powerful island model that allows turning any number of computers connected to the Internet into a cluster of machines. Experiments show that the published linear and supra-linear performance of EAs algorithms can be achieved on both a benchmark and a real-world problem that a run on 20 machines contributed to solve. The architecture used to obtain these results is close to that of Tianhe-1a, China's current fastest supercomputer that yields a theoretical peak performance of 4.7 PetaFlops, thanks to 14336 2.93GHz CPUs and 3 million 575Mhz GPU cores. This paper shows that Evolutionary Computation can not only benefit but also exploit such machines, which prefigure our future desktop computers (Intel has announced that future CPUs would include a large number of GPU cores). It is therefore of the upmost importance to get ready for change and work on platforms such as EASEA, which will allow EC to keep being generic optimizers for such computers.

REFERENCES

Adamidis, P., 1994, Parallel Evolutionary Algorithms: A Review, *4th Hellenic-European Conference on Computer Mathematics and its Applications (HERCMA '98)*, Sept. 1998, Athens.

Alba, E., J. M. Troya, 1999, An analysis of synchronous and asynchronous parallel distributed genetic algorithms with structured and Panmictic Islands, San Juan PR., *Lecture Notes in Computer Science*, Springer-Verlag London, UK, 1586: 248-256

Ausfelder, F., L. A. Baumes, D. Farrusseng, 2011a, Last Developments in Combinatorial Catalysis Research and High-Throughput Technologies, *Catalysis Today*, 159(1), 1-150.

Ausfelder, F., L. A. Baumes, D. Farrusseng, 2011b, Preface, *Catalysis Today*, 159(1), 1.

Baumes, L. A., A. Blansché, P. Serna, A. Tchougang, N. Lachiche, P. Collet, A. Corma, 2009, Using genetic programming for an advanced performance assessment of industrially relevant heterogeneous catalysts, *Materials and Manufacturing Processes*, 24(3), 282-292.

Baumes, L. A., P. , Collet, 2009, Examination of genetic programming paradigm for high-throughput experimentation and heterogeneous catalysis, *Computational Materials Science*, 45(1), 27-40.

Baumes, L. A., S. Jimenez, A., Corma, 2011a, hITeQ: A new workflow-based computing environment for streamlining discovery. Application in materials science, *Catalysis Today*, 159(1), 126-137.

Baumes, L. A., F. Kruger, S. Jimenez, P. Collet, A. Corma, 2011b, Boosting theoretical zeolitic framework generation for the determination of new materials structures using GPU programming, *Physical Chemistry Chemical Physics*, 13, 4674-4678.

Baumes, L. A., M. Moliner, A. Corma, 2007, Prediction of ITQ-21 Zeolite Phase Crystallinity: Parametric Versus Non-parametric Strategies, *QSAR & Combinatorial Science*, 26(2), 255-272.

Baumes, L. A., M. Moliner, A. Corma, 2009, Design of a Full-Profile-Matching Solution for High-Throughput Analysis of Multiphase Samples Through Powder X-ray Diffraction, *Chemistry - A European Journal*, 15(17), 4258-4269.

Baumes, L. A., M. Moliner, N. Nicoloyannis, A. Corma, 2008, A reliable methodology for high throughput identification of a mixture of crystallographic phases from powder X-ray diffraction data, *Cryst. Eng. Comm.*, 10(10), 1321-1324.

Branke, J., A. Kamper, H. Schmeck., 2004, Distribution of Evolutionary Algorithms in Heterogeneous Networks - GECCO 2004, *Lecture Notes in Computer Science*, 3102, 923-934.

Cantú-Paz, E., 2000, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academy Publishers.

Collet, P., E. Lutton, M. Schoenauer, J. Louchet, 2000, Take It EASEA, *6th International Conference on Parallel Problem Solving from Nature*, PPSN, 891-901.

Corma, A., M. Moliner, J. M. Serra, P. Serna, M. J. Diaz-Cabanas, L. A. Baumes, 2006, A New Mapping/Exploration Approach for HT Synthesis of Zeolites, *Chem. Mater.*, 18, 3287- 3296.

Eldredge, N., S. J. Gloud, 1972, *Models in Paleobiology*. T. Schopf. San Francisco, Freeman, Cooper and Co., 82-115.

Farrusseng, D., L. A. Baumes, C. Hayaud, I. Vauthey, P. Denton, C. Mirodatos, 2002, *The Combinatorial Approach for Heterogeneous Catalysis: a Challenge for Academic Research*, Kluwer Academic Publishers.

Farrusseng, D., L. A. Baumes, C. Mirodatos, 2003*, Data Management for Combinatorial Heterogeneous Catalysis: Methodology and Development of Advanced Tools*, Kluwer Academic/Plenum Publishers.

Gordon, V. S., D. Whitley, A. Bohn, 1992, Dataflow parallelism in genetic algorithms. Parallel Problem Solving from Nature, 2. R. Männer; B. Manderick. Brussels, Elsevier *Science*, 533-542.

Jiang, J., J. L. Jordá, J. Yu, L. A. Baumes, E. Mugnaioli, M. J. Diaz-Cabanas, U. Kolb, A. Corma, 2011, Synthesis and Structure Determination of the Hierarchical Meso-Microporous Zeolite ITQ-43, *Science*, 333, 1131.

Krüger, F., O. Maitre, S. Jimenez, L. A. Baumes, P. Collet, 2010, Speedups between ×70 and ×120 for a generic local search (memetic) algorithm on a single GPGPU chip, EvoApplications 2010, April 7-9, 2010, Istanbul, Turkey, *Lecture Notes in Computer Science*, Springer.

Lin, S.-C., W. F. Punch, E. D. Goodman, 1994, Coarse-grain parallel genetic algorithms: categorization and new approach. Parallel and Distributed Processing, 1994. Proceedings. *Sixth IEEE Symposium on Dallas*, TX, USA, 28-37.

Luong, T. V., N. Melab, E. Talbi, 2010, GPU-based island Model for Evolutionary Algoritms. *GECCO*, Portland, Oregon, USA.

Maitre, O., L. A. Baumes, N. Lachiche, A. Corma, P. Collet, 2009a, Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA, *11th Annual Ge-*

netic and Evolutionary Computation Conference, *GECCO-2009*, Montreal.

Maitre, O., N. Lachiche, P. Clauss, L. A. Baumes, P. Collet, 2009b, Efficient parallel implementation of evolutionary algorithms on GPGPU cards, August 25-28, Euro-Par 2009, Delft, The Netherlands, *Lecture Notes in Computer Science*, Springer.

Munetomo, M., Y. Takai, Y. Sato, 1993, An Efficient Migration Scheme for Subpopulation-Based Asynchronously Parallel Genetic Algorithms*, Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, Morgan Kaufmann Publishers, 649.

Pettey, B., M. Leuze, J. Grefenstette, 1987, A Parallel Genetic Algorithm*, 2nd Int. Conf. on Genetic Algorithms*, Hillsdale, N.J., Lawrence Erlbaum Associates.

Serra, J. M., L. A. Baumes, M. Moliner, P. Serna, A. Corma, 2007, *Zeolite Synthesis Modelling with Support Vector Machines: A Combinatorial Approach, Combinatorial Chemistry & High Throughput Screening*, 10(1), 13-24.

Tanese, R., 1987, Parallel genetic algorithms for a hypercube, *2nd Int. Conf. on Genetic Algorithms*, Hillsdale, NJ, Lawrence Erlbaum Associates.

Whitley, D., S. Rana, R. B. Heckendorn, 1999, The Island Model Genetic Algorithm: On Separability, Population Size and Convergence, *Journal of Computing and Information Technology*, 7, 33-48.

OGÓLNE NARZĘDZIE DO OPTYMALIZACJI NA MASZYNACH GPU W ASYNCHRONICZNYM MODELU WYSPOWYM

Streszczenie

W swojej poprzedniej pracy Autorzy przedstawili wydajną implementację Algorytmów Ewolucyjnych (ang. Evolutionary Algorithms – EA) z zastosowaniem procesorów graficznych (Graphics Processing Units – GPU) do rozwiązywania struktur krystalicznych z mikroporami. Ze względu na skomplikowanie materiałów zeolitycznych oraz ciągłą presję na poprawę efektywności symulacji, w niniejszej pracy zaproponowano asynchroniczny model wyspowy na klastrach maszyn wyposażonych w karty GPU. Jest to najnowszy trend w zakresie superkomputerów oraz obliczeń w chmurze (ang. cloud computing). To ostatnie usprawnienie platformy EASEA (ang. EAsy Specification of Evolutionary Algorithms) łatwa specyfikacja algorytmów ewolucyjnych) pozwala na łatwą eksploatację rozbudowanych systemów (komputerów) masowo równoległych. Pokazano, że można osiągnąć ponad-liniowe przyspieszenie w stosunku do jednej maszyny oraz liniowe przyspieszenie stosując klastery o różnych rozmiarach. Takie implementacje wyspowe dla kilku potencjalnie heterogenicznych maszyn otwiera nowe perspektywy dla różnych obszarów zastosowań, w których czasy obliczeń odgrywają kluczową rolę.

Received: October 5, 2010
Received in a revised form: November 25, 2010
Accepted: December 3, 2010