Variational Method for Image Denoising by Distributed Genetic Algorithms on GRID Environment

F. Cannavó, G. Nunnari
Dip. di Ingegneria Elettrica, Elettronica e dei Sistemi
University of Catania
Catania, Viale Andrea Doria 6 - 95125
Email: {fcannavo, gnunnari}@dees.unict.it

D. Giordano, C. Spampinato
Dip. di Ingegneria Informatica e delle Telecomunicazioni
University of Catania
Catania, Viale Andrea Doria 6 - 95125
Email: {dgiordan, cspampin}@diit.unict.it

Abstract—The aim of this paper is to present a novel distributed genetic algorithm architecture implemented on grid computing by using the G-Lite middleware developed in the EGEE project. Genetic algorithms are known for their capability to solve a wide range of optimization problems and one of the most relevant feature of GAs is their structural parallelism, that fits well the intrinsically distributed GRID architecture. The proposed architecture is based on different specialized autonomous entities able to interact in order to carry out a global optimization task. The interaction is based on exchange of knowledge on the problem and solutions. In this way the main problem can be solved by using many cooperative small entities that can be classified into different specialized families that cover only one aspect of the global problem. The topology is based on archipelagos of islands that interact by chromosomes migrating with an user-definable strategy. Grid has been mainly used in the high performance computing area. The properties of the proposed GAs architecture and its related computing properties have great potential in solving big instances of optimization problems. Furthermore this implementation (distributed genetic algorithms with grid computing) is suitable to solve time consuming problems by reducing by executing different instances on many virtual organizations (VOs) according to the Grid philosophy. The proposed parallel algorithm has been tested on denoising problems applied to image processing which are known to be time consuming. The paper reports some results about the time performance compared to traditional denoising filter algorithms.

I. INTRODUCTION

In recent years the increase in the computational power of computers has opened new perspectives in many scientific fields, in particular for optimization tasks. Optimization remains one of the task that cannot be set aside for many engineering applications. Evolutionary algorithms are powerful optimization methods based on the mechanics of natural evolution [1]. It is known that these kind of algorithms can find the global optimum even when there are several local optima. Among the optimization algorithms based on evolutionist theory, the Genetic Algorithms (GA) are unanimously considered a valid procedure for global optimization for a wide range of problems and disciplines. In addition to a consolidated base of existing applications, a huge literature is available that demonstrates the GA efficiency and validity also from the theoretical point of view, however the execution of a GA is generally much demanding of calculation and memory resources. Therefore genetic algorithms are very useful, but they are affected by some problems. One of the problems is the high number of iterations needed. Hence, to solve practical problems by genetic algorithms they should be speeded up somehow. This can be obtained by using some parallelization strategy. Indeed the unique power of evolutionary algorithms shows up with parallel computers. Wright [2] has argued that the best way to avoid being hung up on a poor local optimia is to have the population broken up into many nearly isolated subpopulations. Among several classes of parallel genetic algorithms, one is the so called distributed genetic algorithms (DGA). In distributed genetic algorithms, the entire population is divided into small groups which are called sub populations or islands. It is reported that distributed genetic algorithms have some advantages compared to genetic algorithms with single populations [3] not only the reduced run time [4],[5] but also better reliability, flexibility and accuracy [6]. Moreover several studies concerning with distributed genetic algorithms have clarified that DGAs can find the optimum with small population size and short calculation times [7],[8].

Distributed genetic algorithms can solve large optimization problems provided they can make use of their inherent scalability. There are some researches that focus on the distributed genetic algorithms on clusters of computers for optimization problems. On the other hand, there are very few studies focused on distributed genetic algorithms on Grid computing. In this paper, a novel model of distributed genetic algorithm for grid is proposed and tested for a very time consuming optimization problem concerning image “salt and pepper” denoising. The proposed model follows some guidelines already known in the literature [10][11] and introduces some improvements to exploit the grid features for scalability. The model is based on splitting of the entire population of solutions. The effectiveness of the proposed model is discussed through a numerical optimization problem regarding an image enhancement operation (noise filtering) that is known for
The development of the Grid has opened up new perspectives which could lead to a dramatic increase in the performance of DGA, in terms of execution time and problem size. However, the execution and development of Grid applications requires a high level of expertise and a significant amount of effort. The proposed model is a parallel optimization algorithm which combines the computational speed of grid infrastructure and the software speed of intelligent parallel searching. In the DGA model the intrinsic parallelism of GA has been exploited to speed up the search of the optimum, by using distributed topologies named islands. For its simplicity DGA has been widely used in many parallel computational resources [9]. In the DGA model, more GA evolves independently exchanging elements of the population so that the best solutions migrate in all the islands. In this way the search is quickly steered toward the global optimal solution. In the DGA the synchronization among several islands, as their same existence, is not critical therefore it is possible to have communications also on a not so fast and reliable channel or to foresee that an island can disappear without warning as a result of some technical problem due to the node in which it executes. The level of parallelism of a GA is virtually limitless, therefore, generally, the number of islands on which the DGA evolves is set up as a function of the number of available CPUs. As is largely demonstrated [12] an high number of nodes in which the DGA evolves, also with functions objective not parallelized, the time of search of the solution is drastically reduced. The computational grid gives to the customers huge resources for computation and storage that can be used for evolution of a DGA. An evolving DGA on a grid can be addressed as GDGA (Grid Distributed Genetic Algorithm). In this paper we propose a robust architecture of GDGA that allows to maximize the characteristics of flexibility and scalability of the DGA in a complex environment such as GRID.

II. DISTRIBUTED GENETIC ALGORITHMS ON GRID

The parallization strategies of genetic algorithms can roughly be divided into two main approaches depending on the idea of parallelize evaluation of the fitness function or population respectively, [1]. Our approach is based on parallelizing the evolution of the population. In particular, a difference with classic DGA is that the islands are organized into Archipelagos and a solution sharing is performed to improve the results. In archipelagos, different genetic algorithms are run separately in different islands. After some generations, some individuals in each island are chosen and sent to another island within the same archipelago, perfomed by JAVA MPI. This operation is called migration. The number of individuals that migrate to another islands is another parameter that is called a migration rate.

In this paper, a different approach of distributed genetic algorithms in time consuming optimization problems is proposed. In this approach, the island model is taken for a distributed genetic algorithm and a solutions sharing is performed to improve the results. A general schema of the proposed distributed algorithm on grid is shown in Fig. 1. Distributed genetic algorithms find in the grid solution an optimal environment to be implemented, developed and executed. The main difficulties arise from the characteristics of the Grid itself, namely: complexity, heterogeneity, dynamism and high fault rate. To overcome these difficulties, we have developed an application by using the G-Lite framework (middleware developed in the EGEE Project [13]) that allows an easier and more efficient execution of jobs on a dynamic Grid environment. The Archipelago component automatically performs all the job scheduling steps (resource discovery and selection, job preparation, submission, monitoring, migration and termination) also providing fault recovery mechanisms, and adapts job execution to the changing Grid conditions. On the other hand, the Grid lacks of a standard programming paradigm to port existing applications among different environments. The prototype is implemented in JAVA™.

Fig. 1. Flow Diagram for distributed genetic algorithms.
any central control. This leads to a more flexible management of asynchronous entities on GRID.

Fig. 2. Distributed genetic algorithms on GRID

The components for the proposed architecture are divided into computation units and managing units. The formers perform the optimization algorithm while the latters manage the distribution of the algorithm.

Fig. 3. The proposed architecture for distributed genetic algorithms on GRID

The general architecture scheme is shown in Fig. 3. Blocks in each level are independent by each others. Fig. 4 shows the activation sequence of the jobs whereas the tasks for each block are described in the following.

A. Grid Gateway

The Grid Gateway is the interface with the middleware of the used grid. By implementing the methods of the abstract class of Grid Gateway it is possible to interface to various middleware. The functions carried out by this component are the following:

- Grid Access.
- Job Submission.
- Result Furnish.
- Fault Managing.

The Grid Gateway component has been implemented for G-LITE middleware available in the VO Gilda in INFN [14].

B. Island

The Job Island is the component that is scheduled and executed inside of the GRID and it is responsible for:

- Managing of instances of genetic algorithms with variable genetic parameters.
- Evaluating the fitness function.

The implemented version is an executable command line that accepts the parameters in input as well as the file with the starting population.

C. Archipelago

The Archipelago is the crucial component of the proposed architecture. It is responsible for:

- Submitting the jobs to its grid gateway.
- Managing the islands.
- Managing the possible faults of the islands.

It adopts the following strategies:

- Strategy for the dimensioning of population in the islands.
- Strategy for the migration of chromosomes among the islands.
- Strategy for choosing the genetic operators for the various jobs.

These strategies can depend on the optimization problem under examination. In our case the choices have been:

- All initial populations have the same dimension.
- The genetic operators are chosen in accordance with some fuzzy rules.

D. World

The World component has the object to coordinate the various archipelagos. It performs the following tasks:

- Chromosomes migration between archipelagos.
- Results reporting.

In the proposed architecture, genetic operators are performed in each island. Those are crossover and selection. Taking into account the above considerations the proposed approach cannot be considered as a fully connected multi island genetic algorithm but an asynchronous cooperative multi island genetic algorithm. In this approach only the islands of an archipelago exchange individuals through the archipelago after a parameterizable number of generations. It implies a low overhead and a low network effort. The initial population is uniformly distributed among the available number of islands, and then a sequential GA is locally executed over each subpopulation. The resultant subpopulations are transferred back to the archipelago, and worst individuals of each subpopulation...
are exchanged with the best ones of the rest. Finally, a new population is generated to perform the next iteration sending the mixing chromosomes among the islands. However, the previous algorithm may incur in performance losses when the relative computing power of the nodes involved in the solution process greatly differs, since the iteration time is determined by the slowest machine. In order to prevent these situations we allow an asynchronous communication pattern between islands. In this way, information exchange only occurs between a fixed number of islands, instead of synchronizing the execution of all subpopulations. The minimum number of islands that should communicate in each iteration depends strongly on the numerical characteristics of the problem. We will refer to this characteristic as dynamic connectivity, since the units that exchange individuals differs each iteration.

The proposed algorithm has the following advantages. First, with this algorithm, an efficient sharing can be performed. Therefore, the solutions have high diversity with respect the centralized GAs due to the separate evolving of populations. Secondly, load balance in parallel computing can be performed automatically. Thirdly, the setting of the sharing parameter can be done in each island. The proposed algorithms is examined and discussed with a numerical example in the following sections taking into account an image enhancement optimization problem.

III. IMAGE DENOISING ALGORITHM

In the last decades, the field of image processing became more interesting, sustained by the continuous advance in electrical and computer engineering. The increasing of the computing (processing) power allows to extend the number of applications in this field. The typical image processing system consists of several building blocks that performs different tasks. Among them, noise removal is one of the most common encountered processing steps. Although there are no exact boundaries between the image enhancement and the filtering, the common interpretation considers the filtering operation as responsible for the noise removal. Of course, the image denoising must take into account the noise distribution. A typical noise for the image is represented by the “salt and pepper noise”. It is caused by malfunctioning pixels in camera sensors, faulty memory locations in hardware, or transmission in a noisy channel[19]. For images corrupted by salt-and-pepper noise (respectively random-valued noise), the noisy pixels can take only the maximum and the minimum values (respectively any random value) in the dynamic range.

The median filter was once the most popular nonlinear filter for removing impulse noise, because of its good denoising power [20]. However, when the noise level is over 50%, some details and edges of the original image are smeared by the median filter with a kernel 3x3 as shown in Fig.5. For the edge preserving Nikolova, in [17], has proposed the variational method, which proceeds by minimizing a functional, referred to as energy, that depends on the image and its space derivatives (gradient). Our considered functional is given by the sum of two terms: one represents the deviation from a data image \( y \), which may be marred by noise, and the other incorporates the variation of a function that penalizes oscillations and irregularities, although does not remove high level discontinuities. These discontinuities are considered necessary to preserve the sharpness of the image. Generally an iterative method, related to percentage of noise, is used for the functional minimization, [18], so that the convergence rate depends on the image smoothness.

Some authors (i.e. [15] and [16]) have proposed the genetic algorithms for the image denoising by non-linear filters. In this paper, to preserve the details and to reduce the computational time, we propose a two-stage scheme filter which combines the variational method, the distributed genetic algorithms with a non linear filter based on neural networks and fuzzy logic, this latter introduces a decision stage for the noise detection [21]. In the second stage we use the distributed genetic algorithm proposed in sect. II to solve the optimization problem given by the minimization of the variational functional. More precisely, the noise candidates are first identified by using a neural network that aims to approximate the 2D function represented by the input image \( y \), and then these noise candidates are selectively restored using an objective function with a data-fidelity term and an edge-preserving regularization term. Since the edges are preserved for the noise candidates, and no changes are made to the other pixels, the performance of our combined approach is much better than that of either one of the methods. The two stages of the algorithm are:

- **Noise Detection**: Denote by \( \tilde{y} \) the approximated image obtained by applying the neural network, and \( y \) is the original corrupted. The noisy pixels take their values in the range \( A \) of overall pixels of the image. We define
the noise candidate set as:

\[ N = \{(i,j) \in A : \tilde{y}_{i,j} - y_{i,j} > T\} \]

where T is the threshold computed by using a fuzzy module as is described in [21]. The set of all uncorrupted pixels is \( N^c = N/A \), where A is the set of all the pixels and N is the set of the noisy pixels.

• (Replacement): Since all pixels in \( N^c \) are detected as uncorrupted, we naturally keep their original values. Let us now consider a noise candidate, say, \((i,j) \in N\). Each one of its neighbors \((m,n) \in V_{i,j}\) is either a correct pixel or is another noise candidate, i.e., \((m,n) \in N\), in which case its value must be restored. The neighborhood \( V_{i,j}\) of \((i,j)\) is thus split as \( V_{i,j} = (V_{i,j} \cap N^c) \cup (V_{i,j} \cap N)\). Noise candidates are restored by minimizing the functional, restricted to the noise candidate set \( N\):

\[ F_y|_N(u) = \sum_{(i,j) \in N} \left[ |u_{i,j} - y_{i,j}| + \frac{1}{\beta}(S_1 + S_2) \right] \tag{1} \]

where

\[ S_1 = \sum_{(m,n) \in V_{i,j} \cap N^c} 2 \cdot \varphi(u_{i,j} - y_{m,n}) \]
\[ S_2 = \sum_{(m,n) \in V_{i,j} \cap N} \varphi(u_{i,j} - u_{m,n}) \]

The restored image \( \hat{x} \) with indices \((i,j) \in N\) is the minimizer of the previously functional which has been obtained by using Distributed Genetic Algorithms onto \( N \) instead of onto \( A \).

IV. SIMULATION

In this section we evaluate the performance of the Grid oriented Distributed Genetic Algorithm described in previous sections, in the solution of an image denoising problem. In our case we consider an initial population of 500 individuals. The sequential GA executed on each island performs a fixed number of iterations (50), with a mutation and crossover probabilities in \([0.1\%-0.2\%]\) and \([50\%-70\%]\), respectively. The exchange probability of best individuals between islands is 10%. Among the commonly tested 512-by-512 8-bit grayscale images, with homogeneous region, the Lena image is 10%. Among the commonly tested 512-by-512 8-bit gray-scale images, with homogeneous region, the Lena image shown in Fig.5(a) has been selected for our simulations. Its dynamic range is \([0,255]\). In the simulations, images have been corrupted by "salt" (with value 255) and "pepper" (with value 0) noise with equal probability. Also a wide range of noise levels varied from 10% to 90% with increments of 20% have been tested. Restoration performances are quantitatively measured by the cpu-time (in seconds), the peak signal-to-noise ratio (PSNR), the mean absolute error (MAE) defined as follows:

\[ \text{PSNR} = 10 \cdot \log_{10} \frac{255^2}{\frac{1}{MN} \sum_{i,j} (r_{i,j} - x_{i,j})^2} \]
\[ \text{MAE} = \frac{1}{MN} \sum_{i,j} (r_{i,j} - x_{i,j})^2 \]

where \( r_{i,j} \) and \( x_{i,j} \) denote the pixel values of the restored image and the original image, respectively. The edge-preserving function \( \varphi(t) = |t|^{1/3} \) will be used. That leaves only the parameter \( \beta \) to be determined, that we fix equal to 5 in all the tests. The proposed filter has been compared with other wellknown filters, which are the ones proposed by the authors in [21] and the ones like the median filter [19], PSM filter [22] and the Chan’s filter[23]. For median filter, the window sizes are chosen for each noise level to achieve its best performance. The decision threshold in PSM is also tuned to give the best performance in term of PSNR. We summarize the performance of different methods in Tables I, II, III. From the tables, we can see that all the methods have similar performance when the noise level is low.

![Table I](image)

<table>
<thead>
<tr>
<th>Noise</th>
<th>Median</th>
<th>Neural</th>
<th>PSM</th>
<th>Chan</th>
<th>Prop.Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>35.497</td>
<td>37.881</td>
<td>30.749</td>
<td>41.909</td>
<td>40.286</td>
</tr>
<tr>
<td>30%</td>
<td>32.155</td>
<td>35.537</td>
<td>28.245</td>
<td>36.075</td>
<td>35.589</td>
</tr>
<tr>
<td>50%</td>
<td>31.811</td>
<td>33.410</td>
<td>27.522</td>
<td>33.085</td>
<td>31.586</td>
</tr>
<tr>
<td>70%</td>
<td>18.998</td>
<td>22.589</td>
<td>21.831</td>
<td>29.705</td>
<td>27.685</td>
</tr>
</tbody>
</table>

![Table II](image)

<table>
<thead>
<tr>
<th>Noise</th>
<th>Median</th>
<th>Neural</th>
<th>PSM</th>
<th>Chan</th>
<th>Prop.Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>4.1671</td>
<td>1.1315</td>
<td>3.7995</td>
<td>0.40465</td>
<td>0.51813</td>
</tr>
<tr>
<td>30%</td>
<td>7.2887</td>
<td>3.5126</td>
<td>9.0002</td>
<td>1.571</td>
<td>1.8545</td>
</tr>
<tr>
<td>50%</td>
<td>19.4539</td>
<td>15.5269</td>
<td>9.2462</td>
<td>2.9037</td>
<td>2.9708</td>
</tr>
<tr>
<td>70%</td>
<td>49.5151</td>
<td>45.5508</td>
<td>15.7796</td>
<td>4.9832</td>
<td>5.0505</td>
</tr>
<tr>
<td>90%</td>
<td>99.8366</td>
<td>93.6619</td>
<td>64.6212</td>
<td>11.6111</td>
<td>11.7555</td>
</tr>
</tbody>
</table>

![Table III](image)

<table>
<thead>
<tr>
<th>Noise</th>
<th>Median</th>
<th>Neural</th>
<th>PSM</th>
<th>Chan</th>
<th>Prop.Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.125</td>
<td>764.109</td>
<td>11.4375</td>
<td>1859.890</td>
<td>443.524</td>
</tr>
<tr>
<td>30%</td>
<td>0.056</td>
<td>754.4062</td>
<td>10.5312</td>
<td>9827.468</td>
<td>1062.345</td>
</tr>
<tr>
<td>50%</td>
<td>0.03125</td>
<td>768.818</td>
<td>19.375</td>
<td>22901.297</td>
<td>4567.281</td>
</tr>
<tr>
<td>70%</td>
<td>0.03125</td>
<td>765.819</td>
<td>12.046875</td>
<td>50193.141</td>
<td>9543.609</td>
</tr>
<tr>
<td>90%</td>
<td>0.03125</td>
<td>769.296</td>
<td>13.484735</td>
<td>100992.916</td>
<td>15432.95</td>
</tr>
</tbody>
</table>

As is shown in the tables above the proposed filter has a good performance in terms of PSNR and MAE, and is very efficient in term of CPU-TIME respect to the Chan's Filter that is considered the best filter in literature for the "salt and pepper" denoising. Fig.6(b) shows the restored Lena image, when the original image is corrupted with 70% of noise, 6(a). The architecture of the GDGA consists of four archipelagos, each one with five islands. For our simulations we use G-lite Middleware for the GRID, where the user need of a "proxy certificate" able for the User Interface Machine. To submit our job it is necessary to use the command edg-job-submit with a...
jdfile that have to contain all the attributes for the execution of the job, where is specified the needed resources for the execution.

Fig. 7 shows the status (and so the time needed for the elaboration) for a simulation with 10% of noise with Lena image 256x256 obtained using the command edg-job-status.

V. CONCLUSION

In this paper, we propose a novel distributed genetic algorithm architecture implemented on grid computing, for details preserving restoration method in salt and pepper image denoising. It is one of the best filter for removing salt-and-pepper noise. In fact experimental results show that our method represents the best trade-off between the PSNR and MAE with the CPU-TIME results. Future works about the image denoising is toward a well filter decision based for the noise detection, maybe based on Bayesian classifier provided it lend itself toward GRID implementation. About the distributed genetic algorithm on grid, the next step is the parallelization of the fitness function, when possible, evaluation which, at present, represents a slowing down critical point.

REFERENCES


[14] https://gilda.cnfin.it/


[23] Chan, R.H., Ho C.W. Nikolova M. Salt and pepper noise removal by median type noise detectors and detail-preserving regularization, IEEE Transactions on Image processing, 2005