International Journal of Modern Physics C, Vol. 12, No. 1 (2001) 1–16

OPTIMIZATION OF A DISTRIBUTED GENETIC ALGORITHM ON A CLUSTER OF WORKSTATIONS FOR THE DETECTION OF MICROCALCIFICATIONS

A. BEVILACQUA

Department of Electronics, Computer Science and Systems
University of Bologna, viale Risorgimento
2, 40136 Bologna, Italy
National Institute for Nuclear Physics (INFN) viale Berti Pichat
6/2, 40127 Bologna, Italy
E-mail : bevila@bo.infn.it

R. CAMPANINI and N. LANCONELLI

Department of Physics, University of Bologna and National Institute
for Nuclear Physics (INFN) viale Berti Pichat
6/2, 40127 Bologna, Italy

Received 2 January 2001
Revised 30 January 2001

We have developed a method for the detection of clusters of microcalcifications in digital mammograms. Here, we present a genetic algorithm used to optimize the choice of the parameters in the detection scheme. The optimization has allowed the improvement of the performance, the detailed study of the influence of the various parameters on the performance and an accurate investigation of the behavior of the detection method on unknown cases. We reach a sensitivity of 96.2% with 0.7 false positive clusters per image on the Nijmegen database; we are also able to identify the most significant parameters. In addition, we have examined the feasibility of a distributed genetic algorithm implemented on a nondedicated Cluster Of Workstations. We get very good results both in terms of quality and efficiency.

Keywords: Genetic Algorithm; Distributed Processing; Parallel Processing; Cluster Of Workstations; Image Processing; Computer Aided Diagnosis; Digital Mammogram; Microcalcifications.

1. Introduction

Breast cancer is the most common form of cancer among women. The presence of microcalcifications in breast tissues is one of the main features considered by radiologists for its diagnosis. In order to reduce detection errors, one of the possible solutions is to assist doctors with a computer aided system. The computer output is presented to radiologists as a "second opinion" and can improve the accuracy in the detection task.[1] We have developed a Computer Aided Diagnosis (CAD)

scheme for detecting clustered microcalcifications in digital mammograms.[2,3] Our scheme contains a large number of parameters such as filter weights, threshold levels and feature ranges for false-positive reduction steps. The performance of the system depends strongly on the settings of these parameters. Hence, it is essential to make an accurate choice of them in order to obtain the best results. The number of these parameters is large (about thirty) and therefore it is difficult to get the optimal performance especially since the values of some parameters are correlated with each other.

In this paper, we present an automated method for the selection of the parameter values by means of a genetic algorithm (GA). In a previous study, the selection was performed manually;[2,3] we refer to this procedure as the "hand tuned" one. GAs have been successfully applied to many diverse and difficult optimization problems.[4–8] GAs search the solution space to maximize (minimize) a fitness (or cost) function by using simulated evolutionary operators such as mutation and sexual recombination. In this study, the fitness function to be maximized reflects the goal of maximizing the number of true-positive detections while minimizing the number of false-positive alarms. GA utilization requires the determination of several issues: cost function design, parameter set representation, population initialization, choice of selection function, choice of genetic operators (reproduction mechanisms) for evolution, and identification of termination criteria. In a number of applications where the search space was too large for other heuristic methods or too complex for analytic treatment, GAs produced favorable results. Other researchers have shown that GAs could improve the performance of a CAD scheme.[9,10]

We will evaluate how the fluctuations in the parameter values influence the performance of the CAD scheme and which parameters affect most the cost function. Our goal is as well to select, by using a GA, the most significant parameters; we then can also study easily the effects of additional parameters as they are considered in the optimization due to the excellent scalability of GAs. It would be impossible to perform manually these tasks because of the correlations among the parameters. The GA needs to be run repeatedly in order to obtain a good optimization. Due to the very long time required for one run, it would be impracticable to execute the GA on a sequential architecture. We therefore implement a distributed GA on a Cluster Of Workstations (COWs) by realizing a global parallelized GA. In this type of parallel GAs, there is only one population as in the serial GA, and even if the evaluation of individuals is parallelized explicitly, the algorithm remains unchanged. In this way, we could apply easily the existing principles for sequential GAs.

We accomplish the optimization of our CAD scheme by using a database of 40 digitized mammograms coming from the Nijmegen hospital: this database is considered as a benchmark for CAD systems. Performances of the detection scheme are shown by means of Free Response Operating Characteristic (FROC) curves. A jackknife statistical test is performed to evaluate the average performance of our

method on unknown cases. This assessment of a generalization of the CAD scheme was not feasible previously because of the large amount of time needed for the manual optimization.

2. Domain Problem: The Detection Scheme

Microcalcifications are very small spots which appear brighter than the surrounding normal tissue. Typically, they are between 0.1 mm and 1 mm in size and are of particular clinical significance when found in clusters of five or more in a 1 cm^2 area. Most of the clusters consist of at least one evident microcalcification and other more subtle signals. Our approach to the detection task includes two different methods: the first one (coarse) is able to detect the most obvious signals while the second one (fine) based on multiresolution analyses, discovers less visible microcalcifications.[2] Signals coming out from these two methods are combined through a logical OR operation and then clusterized to give the final result. In both methods (coarse and fine one), there are some common steps,

- pre-processing which isolates breast tissue,
- filtering in which structured background is removed,

Table 1. Parameters used in the optimization process.

Parameters of the coarse method	Parameters of the fine method
Size of the local thresholding window	
Threshold for Gaussianity test h_t	
Values for local thresholding k	
Minimum edge gradient (EG)	
Maximum EG	
Minimum average local gradient (ALG)	Minimum neighbor gray level (NGL)
Maximum ALG	Maximum NGL
Minimum area of signal	Minimum area of signal
Maximum area of signal	Maximum area of signal
Minimum gray level (GL)	Minimum GL
Maximum GL	Maximum GL
Minimum degree of linearity (DL)	Minimum DL
Maximum DL	Maximum DL
ct_{11}, in: $\mathrm{GL} > ct_{11} * \mathrm{EG} + ct_{12}$	p_1, in: $\mathrm{EG} < p_1 * \tanh(p_2 * \mathrm{GL})$
ct_{12}, in: $\mathrm{GL} > ct_{11} * \mathrm{EG} + ct_{12}$	p_2, in: $\mathrm{EG} < p_1 * \tanh(p_2 * \mathrm{GL})$
ct_{21}, in: $\mathrm{DL} < ct_{21} * \mathrm{ALG} + ct_{22}$	p_3, in: $\mathrm{EG} > p_3 * \mathrm{GL} + p_4$
ct_{22}, in: $\mathrm{DL} < ct_{21} * \mathrm{ALG} + ct_{22}$	p_4, in: $\mathrm{EG} > p_3 * \mathrm{GL} + p_4$
ct_{31}, in: $\mathrm{DL} > ct_{31} * \mathrm{ALG} + ct_{32}$	p_5, in: $\mathrm{GL} > p_5 * \mathrm{NGL} + p_6$
ct_{32}, in: $\mathrm{DL} > ct_{31} * \mathrm{ALG} + ct_{32}$	p_6, in: $\mathrm{GL} > p_5 * \mathrm{NGL} + p_6$

- signal extraction to find out signals due to microcalcifications candidates,
- false positive reduction where microcalcifications are separated by calculating a set of features[2,11] from false signals.

In all these tasks, there are several parameters to be tuned; we used thirty-three of them (listed in Table 1) for the optimization process with the GA. Most of them are thresholds used to choose the range of feature values in the false positive reduction phase; others are used for selecting regions of interest or extracting signals.[2] Any individuals of the population considered in GA optimization is therefore described by a string of thirty-three values. Each parameter value (gene) can be a real or an integer number according to the domain of the parameter itself. The purpose of the optimization of a CAD scheme is to find out the set of parameters which yields the highest number of true positive clusters of microcalcifications with the lowest rate of false positive clusters (i.e., the best tradeoff between sensitivity and specificity). This tradeoff is controlled by the design of the fitness function.

3. Genetic Algorithm

3.1. *Overview*

GAs are heuristic methods that operate on pieces of information like nature does on genes in the course of evolution. Individuals are represented by a string of letters of an alphabet and they are allowed to replace, crossover and mutate. Replacement simply carries an individual from one iteration to the next; crossover combines two distinct individuals to create a new one while mutation randomly changes the value of one or more parameter values in a given individual. All individuals at one generation are evaluated by a cost (or fitness) function. Depending on the generation replacement mode, a subset of parents and offspring enters the next reproduction cycle. After a number of iterations, the population consists of individuals that are well adapted in terms of the fitness function. The basic outline of a GA is as follows,

(1) initialize a population of individuals,
(2) evaluate all individuals of the population,
(3) generate new individuals through replacement, crossover and mutation,
(4) go back to step 2 until the termination criterion is satisfied.

The advantages in using GAs are that they require no knowledge or gradient information about the response surface, they are resistant to becoming trapped in local optima and they can be employed for a wide variety of optimization problems. On the other hand, GAs could have trouble in finding the exact global optimum and they require a large number of fitness function evaluations. It is very difficult to obtain an analytic relationship between the sensitivity of the CAD and the parameter values to be optimized. However, since a GA does not need this kind of information, it is suitable in our optimization task.

3.2. *Experimental design*

The configuration of a GA needs the investigation of several points. We have already seen in Sec. 2 how to represent the individual of the population. In this section, we will outline our preferences concerning the reproduction mechanisms, the choice of the fitness function and of the termination criterion.

The initial population can be created either randomly or by perturbing an input individual. If there is an explicit knowledge about the system being optimized, that information can be included in the initial population. The initialization procedure should not be critical as long as the initial population spans a wide range of variable settings.

An evolutionary strategy needs to be adopted in order to generate individuals for the next generation. We choose an *elitist generation* replacement as replacement operator. Namely the individuals are ranked by their fitness and only the best of them (usually 10% of the population) are taken unchanged into the next generation. In this way, we guarantee that good individuals are not lost during a run. Other children are coming from crossover and mutation. Crossover operator acts on pairs of individuals (parents) to produce new strings (offspring) by exchanging segments from the parents' strings. The probability that the parents are recombined is a user-controlled option (p_{CO}) and frequently set to a high value. If the parents are allowed to mate, the crossover operator is employed to exchange genes, otherwise, the parents are placed into the next generation unchanged. Traditionally, the two most common are the one-point and two-point crossover operators. In the one-point method, a crossover point is randomly selected along the string and the genes up to that point are swapped between the two parents. Analogously, more than one crossover point can be selected and only the fragments between those positions exchanged (n-point crossover). When the number of crossover points is equal to the number of genes, we have the so-called uniform crossover. The mutation operator simply randomly changes the value of a gene. Usually, the probability of performing this variation (p_{MUT}) is very low.

The aim of the cost function is to encode numerically the performance of an individual. In our case, a pair (true positives, false positives) is mapped by this function to a real number normalized between 0 and 1. That number represents the excellence of a pair obtained by a particular individual (i.e., a set of parameters in the CAD scheme). We designed the cost function as a 2D Gaussian with the maximum in the most desired point (100% of true positives and 0 false positives). The "hand tuned" method yields a result of 91.4% true positives and 0.4 false positives per image (*fpi*).[2,3] Since the goal of the optimization with the GA is to improve this performance, we plot the fitness function with a rapid slope around 90% true positives and 0.5 *fpi*; in this way, the convergence towards a number of true positives greater than 90% and a number of *fpi* smaller than 0.5 will be a benefit. The fitness function used is depicted in Fig. 1.

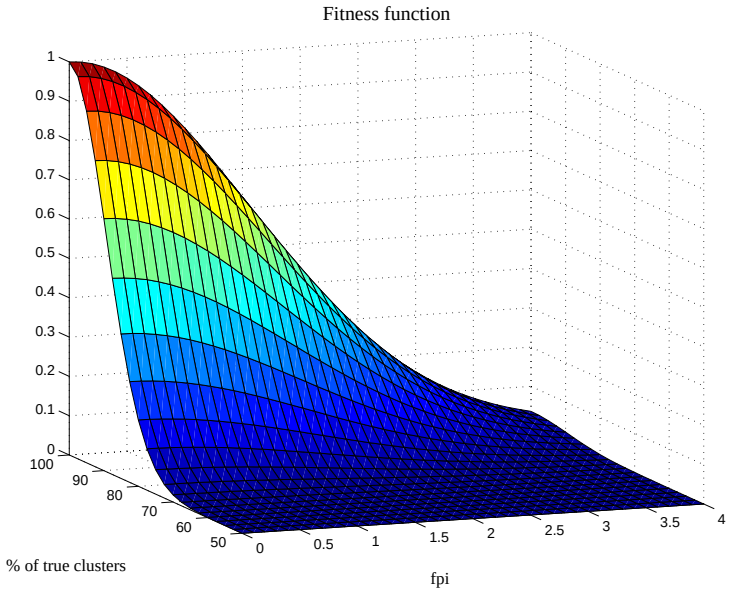

Fig. 1. The fitness function.

In order to end the evolution of the population, we must choose a termination criterion. We test some of them,

(1) fixed number of generation,
(2) average of the fitness of the entire population,
(3) average of the fitness of the best individuals.

Regarding points number 2 and 3, we stop the evolution when the average has reached a plateau. The final result of the GA optimization is the best individual of the last iteration.

4. Implementation

4.1. *Sequential algorithm*

The kernel of the sequential GA described in Sec. 3 is using pseudo-code,

1. <Create first random population>
2. <Fitness evaluation>
3. WHILE <Termination condition is false> DO
4. <Replacement>
5. <Crossover>
6. <Mutation>
7. <Fitness evaluation>
8. ENDWHILE

The GA supervises the executions of the pre-existing "basic program"[2,3] which solves the domain problem and provides fitness evaluation. We want to point out that there are two hidden *for* loops. The first regards the evaluation step in code line 2 (and 7) since the problem is solved independently for each individual of the population. The second is inherent to the "basic program", the detection algorithm which follows the scheme described in Sec. 2 over a whole database of mammographic images: the evaluation of one individual requires the independent execution of the detection program *for each* image in the database. These loops will be exploited in the parallel development of the algorithm.

4.2. *Parallel algorithm*

GAs running on parallel or distributed architectures constitute the class of Parallel Genetic Algorithms (PGAs). PGAs can be classified in three basic groups according to the structure of parallelization: *global*, *coarse-grained* and *fine-grained*.[12,13] In global parallelization, any individual can mate with any other because the operators and the evaluation of the individuals are explicitly parallelized often by a "master processor" that sends individuals to other processors for the evaluation and applies genetic operators.[14] In coarse-grained PGAs, a few subpopulations are modeled using a migration operator to move individuals among them. They are usually modeled on distributed memory MIMD computers. Fine-grained parallelism runs on massively parallel computers, SIMD and MIMD, ideally with one individual per processor. Global parallelization is one of the most common ways to realize a parallel GA. The purpose of this type of algorithm consists in dividing the task of evaluating the population among several processors, following the master-slave paradigm, because their fitnesses are independent from each other. The master program stores the entire population and performs an iterative decomposition: on each generation, it sends a fraction of the population (one or more individuals) to each slave processors and waits for the results from the slaves to come back. Slaves are self-scheduled, they ask the master for more work as their task ends. This algorithm behaves in a synchronous manner, since the master waits to receive the fitness values for *all* the individuals, before proceeding into the next generation. Once all individuals have been computed, the master performs replacement, crossover and mutation operations to create a new generation. In this way, the GA operations remain global and the existing design guidelines for simple (sequential) GAs are directly applicable. In our GA, individuals are short strings of bytes and they are not time consuming from a communication point of view. For this reason, GA can also exploit the data parallelism of the detection algorithm. We remark that inside each generation, an image belongs to an individual. Therefore, we can create a new item, called *chunk*, which is constituted by an image identifier and by the individual the image belongs to. The master program on each generation sends a chunk, instead of one individual, to each slave processors. In this way, the maximum idle time of the program shrinks from the time needed for computing a whole database

to the time needed to compute one image. The master code scheme looks like the one of the sequential algorithm listed in the previous subsection, but, code lines 2 and 7 (the fitness evaluation step) become,

```
 9. SEND <one chunk to each slave>
10. WHILE<the generation analysis is not complete> DO
11.    RECEIVE <results from a slave>
12.    IF <there are still chunks> THEN
13.       SEND <one chunk to that slave>
14.    ENDIF
15. ENDWHILE
```

and each slave performs the detection algorithm. "RECEIVE", a blocking message-passing routine, blocks the process until a result has arrived. An asynchronous "SEND" immediately sends the data and computation on the master process resumes as soon as data are safely on their way. In the standard master-slave scheme, workers (slaves) are self-scheduled and the manager (the master) remains idle and ready to satisfy requests as they arrive. Here, we use a modified version of the manager-workers paradigm, the "*working-manager*" model[15] in which the manager uses its idle time to process data itself thus increasing the overall performance.

5. Results

5.1. *Performance analysis*

COWs are a powerful computers ensemble — small Symmetric MultiProcessor (SMP) systems can be found within many of the modern computers — which have replaced single, more expensive parallel machines in academic institutions and industry. The cluster we used is a loosely coupled heterogeneous computer network consisting of six workstations (four SMPs and two single-processor machines with a total of 10 computing nodes) connected to a LAN by a 100 Mbit Ethernet switch. Workstations are listed below according to their performance with the sequential algorithm,

W_1) 1 Pentium III 450 MHz, 512 MB RAM;
W_2) 1 SMP: 2 Pentium III 450 MHz, 256 MB RAM;
W_3) 1 SMP: 2 Pentium II 450 MHz, 512 MB RAM;
$W_{4,5}$) 2 SMPs: 2 Pentium II 400 MHz, 512 MB RAM;
W_6) 1 Mobile Pentium II 366 MHz, 128 MB RAM.

Actually, the better performance of processor W_1 compared to the W_2 machine is due to more advanced technology not to a different amount of memory.

All the code is written in C, *Linux* is the operating system, *PVM* libraries supply the communication routines[16] and *gcc* is the C compiler. Clearly, this is a low-cost cluster with no-cost software.

Since we want also to show that a *nondedicated* COW could improve results in research applications, we ran our jobs while other users were using the machines for normal activities. To get reliable performance measurements, we ran ten times in different periods of the year and we report average values. Anyway, users' "usual" activity yields a roughly uniform mean workload, if considered on sliding windows of at least one day, and there is no substantial difference among our long-time jobs runs.

We adopt static mapping and do not introduce any degree of virtual parallelism, by assigning one task to each processor, for a total amount of 10 processes.

Each population consists of 30 individuals and each generation requires 27 individuals to be computed. Each generation takes about 1 h of elapsed time to be calculated on this cluster.

By means of the Linux OS *times* and *gettimeofday* routines, we get both *CPU time* and *wall clock time* measures necessary to obtain the results.

This data parallel application has a very coarse grain structure and in addition small amount of data are transferred when communication takes place. For this reason, time due to communication between master and slaves is irrelevant and it has not been considered.

We then focus our attention on relative speedup of the whole cluster with respect to each workstation,

$$S_i = \frac{T_s(W_i)}{T_p}\,, \tag{1}$$

where $T_s(W_i)$ is the time it takes to the sequential algorithm on the workstation W_i, and T_p is the time of the parallel algorithm. The upper line in Fig. 2 is the

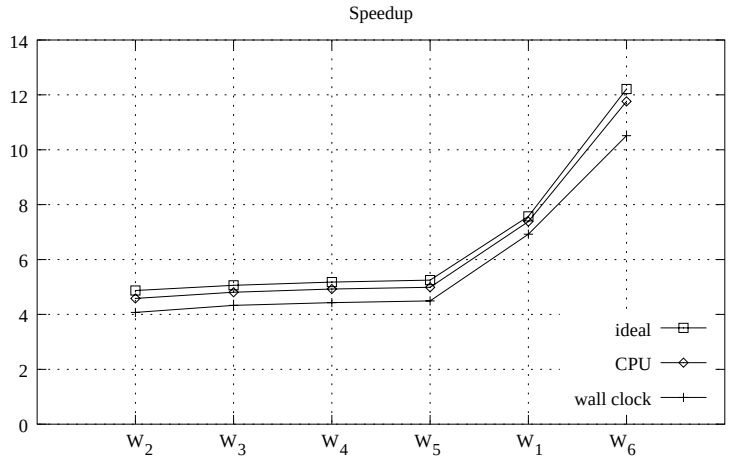

Fig. 2. Relative speedup.

"ideal" (theoretical) speedup I,

$$I_i = \frac{P_T}{P_i}\,. \tag{2}$$

$P_i = T_s(W_1)/T_s(W_i)$ is the *power weight* of W_i compared with that of the fastest workstation W_1. P_T is the power weight of the *whole* cluster and its value is obtained by summing over all P_i (summing twice for SMPs).

This algorithm with *this* ensemble of workstations gives $P_T = 7.57$, i.e., the *power* of this cluster is 7.57 times the one of processor W_1.

Figure 2 shows ideal speedups I_i and speedups S_i computed by measuring CPU time and wall clock time respectively. Along the x axis, workstations (not processors!) are represented according to their increasing performance. Consequently, W_2 SMP is the first instead of the single processor W_1 workstation. Values for ideal speedup range from about 5 to 12: the lower this value, the faster the workstation is. We see that the slope of real speedup lines and the one of the ideal speedup are much alike whether speedups arise from CPU or wall clock time measures. Anyway, W_1 and W_6 values are anomalous and this becomes more evident if we compare ideal and wall clock lines. With regard to W_1, we recall that it is a single-processor machine and it suffers less than SMPs because of the idle time. On the other side, W_6 is single-processor too, but, it lacks memory and this makes its speedup the most different with respect to the ideal one.

The "weighted" efficiency is shown in Fig. 3,

$$\mathrm{Eff}_i = \frac{S_i}{I_i}\,. \tag{3}$$

We observe how the intrinsically parallel nature of this problem has been successfully exploited by dividing original data in chunks. We obtain excellent results in

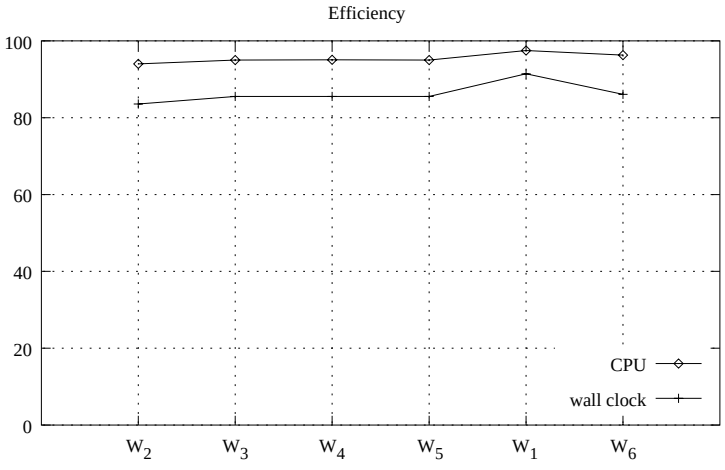

Fig. 3. "Weighted" efficiency.

term of efficiency if we consider CPU time. The algorithm needs synchronization after a whole generation has been computed and this requires more than 1 h of CPU time. In the worst case, we could imagine all workstations but the slowest one end their calculation. The idle time would be about 50 s which is irrelevant. Experiments showed a mean value of about 10 s. On the contrary, time needed to read images is quite significant because this algorithm has low locality. Images have been stored on a server and read via *nfs*. Each image (8 MB) takes about 1.5 s to be read and for each generation, 27×40 images should be read. This sequential step causes a loss in terms of efficiency and it becomes more visible if we consider wall clock time. Since we choose to execute our runs inside a normal working environment, we never had all resources reserved for this application and hence, we could not store in memory all the images but only some of them. Nevertheless, with roughly evenly distributed light loaded workstations, CPU efficiency remains high ranging from about 95% until to 98% for W_1 (see discussion about speedup). W_6 too, the slowest machine, keeps its efficiency high, thanks to its slowness! As expected, efficiency goes down if we measure wall clock time mainly beacuse of swapping and the high number of context switches.

Starting from "hand tuned" results, 87 generations were necessary for the algorithm to converge and it took roughly 4 days (90 h) for each run.

5.2. *Experimental results*

The main goal of the present study is to show that the performance of our CAD detection scheme improves due to the optimization based on the GA. To this end, we depict in Fig. 4 two FROC curves: one related to the "hand tuned" method and one for the optimized scheme. We can see that the sensitivity of the optimized scheme is always larger than in the "hand tuned" case. The GA allows to outperform the previous method by some percents. Even if at first sight it could seem only a

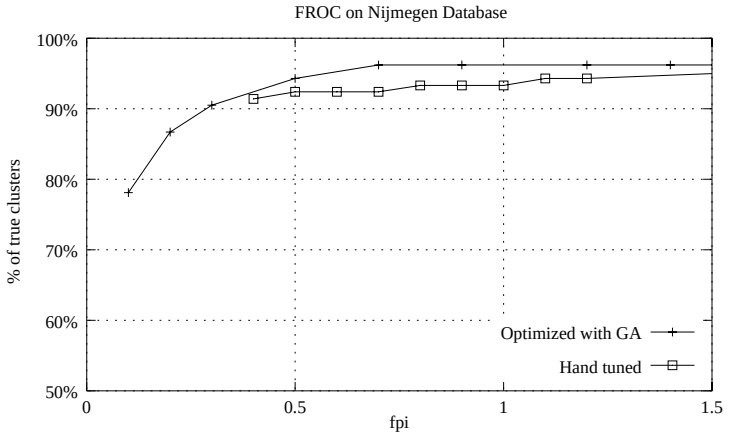

Fig. 4. FROC of the detection methods on the 40 images (Nijmegen) database.

slight improvement, nevertheless, this is extremely important because here it is necessary to minimize the losses of clusters of microcalcifications, maintaining at the same time a low rate of false alarms. Indeed, to avoid any losses of suspect cases is a vital point in issues regarding the detection of lesions for early breast cancer diagnosis; therefore, any small step towards a sensitivity of 100% is crucial. The best solution achieved by the GA is an individual with a fitness value of 0.860 which corresponds to 96.2% of true clusters with 0.65 *fpi*. To obtain these results, we utilize a population of 30 individuals with uniform crossover ($p_{\mathrm{CO}} = 0.8$) and $p_{\mathrm{MUT}} = 0.1$. The convergence of the GA evolution has required the computation of 2352 individuals (corresponding to 87 generations). Let us consider the best individual: focusing our attention on its genes values, we can find out the differences between them and the parameter values of the "hand tuned" results. We can notice that these changes reduce the range of the values which identify the true signals. In particular, regarding the coarse method, the ranges of area, GL, EG and DL are narrower than those achieved in the "hand tuned" study (e.g., the minimum area of signal increases from 3 to 5 pixels whereas the maximum area decreases from 30 to 22). It is possible to observe a similar effect in the fine method as shown in Fig. 5. The curves are described by the parameters p_1 and p_2 (see Table 1) which separate true signals from false detections. Signals above the curve are kept whilst signals below it are eliminated. Also in this case, the GA is more selective in maintaining true signals: if a signal has a given GL, the GA keeps it only if the EG is higher with respect to the "hand tuned" case.

We can therefore summarize that the GA optimization tends to restrict the range of features which characterize the true signals. That allows to cut the number of false positive signals without losing too many true ones.

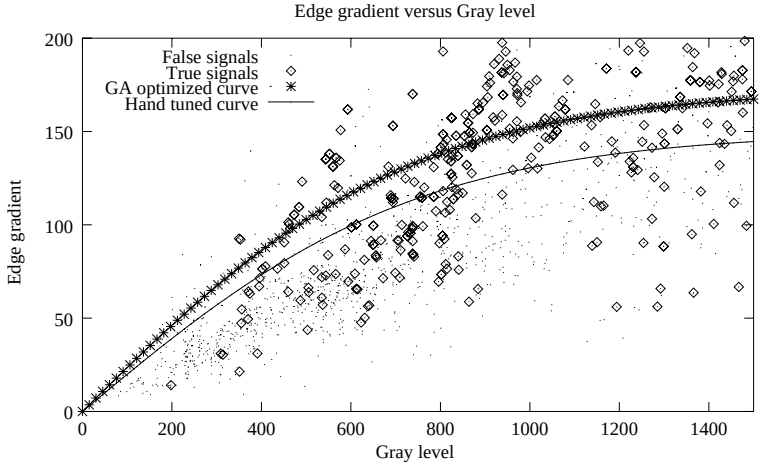

Fig. 5. Scatter plot of EG versus GL. The two curves, described by p_1 and p_2, separate false signals from true ones.

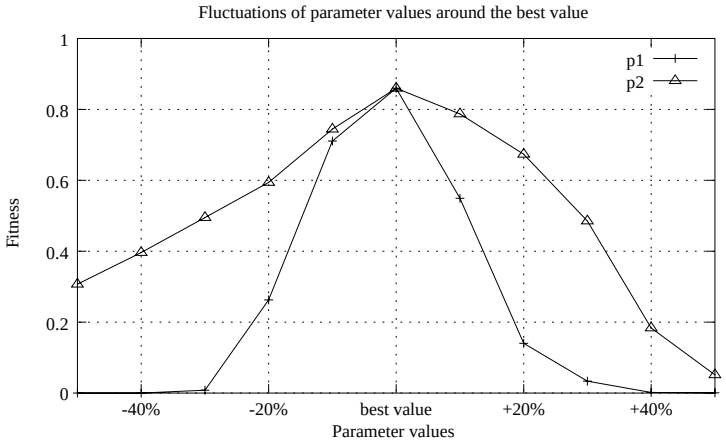

Fig. 6. Variation of fitness due to the change of parameters p_1 and p_2 with the other parameters fixed at their best values.

Another issue investigated is how the fluctuations in the parameter values influence the performance of the CAD system. Starting from the best individual, we vary the value of the first gene around its best value (the one discovered by the GA), keeping fixed the other genes. The variation of the parameter ranges within $\pm 50\%$ its best value. We repeat this process for all the genes, each time maintaining the other values fixed at the best solution. In this way, we can see which parameters affect most the fitness value. In Fig. 6, we can see an example of how the fitness changes due to the variation of p_1 and p_2 separately. We notice that p_1 is a very significant parameter because the fitness goes to zero with a modification of p_1 of only $\pm 30\%$ around its best value. It turns out that the k value for local thresholding (coarse method), p_1, p_2 and the maximum of the DL (fine method) are the most significant parameters. A small fluctuation of their values indeed implies a rapid fall of the fitness. On the other hand, there are some genes which do not affect the value of the fitness. They are, in particular: the maximum of the EG, the minimum and the maximum of the ALG, ct_{31} and ct_{32} for the coarse method and p_4, p_5 and p_6 for the fine method. A variation up to 50% of them around their best value does not cause any change in the fitness value. This fact has allowed us to perform an optimization without these parameters (keeping them fixed at their best value). With this reduced set of parameters (25 genes instead of 33), we obtain the same results of those cited above (fitness value of 0.860) by analyzing only 1731 individuals instead of 2352 (64 generations instead of 87). This happens because with less parameters, the search space of the GA is reduced, resulting in a faster convergence.

We have also tested different termination criteria. The simplest one is to perform a prefixed number of generations. We reject it because it does not take into account the dynamic evolution of the population. It could happen that the interruption of

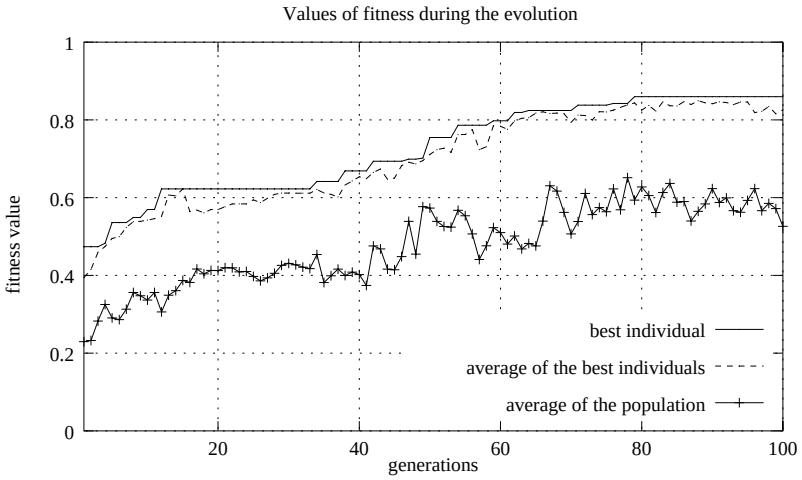

Fig. 7. Values of the fitness of the best individual, average of the best six individuals and average of the population during the evolution of the GA.

the evolution takes place in a phase with a clear improvement of the population, thus, precluding the possibility of further better results. We therefore focus our attention on two criteria which consider the state of the evolution: the average of the fitness of the best individuals and the average of the fitness of the entire population. Their variation and the change of fitness of the best individual during the evolution is shown in Fig. 7. We can see that the trend of the best individual is clearly much more correlated with the value of the best 6 individuals than with the average of the whole population. Hence, it is probable that small fluctuations on the best individuals' fitness would mean the achievement of the best result (in

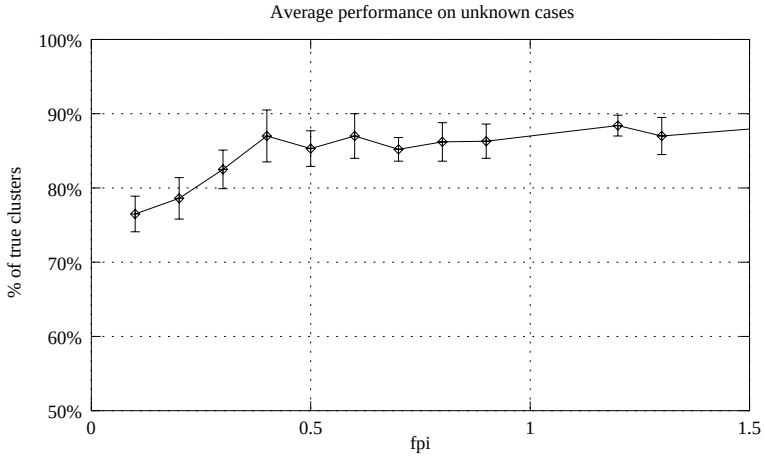

Fig. 8. Average performance of the CAD system on test images.

Fig. 7, this happens after nearly 85 generations). Thus, we stop the evolution of the GA when the fluctuation in the average of the fitness of the best six individuals remains within a small fixed threshold.

Finally, we examine the performance of the CAD system in the presence of cases not used in training. For this purpose, we carry out a jackknife statistical test. In practice, we divide the 40 images into two halves: the training set and the test set. We perform the GA optimization on the training group and then calculate an FROC curve relative to the test images. We repeat this procedure ten times and we average the results of the different test sets. The average performance of the CAD scheme is shown in Fig. 8.

6. Conclusion

The optimization of our detection method, by means of a GA, has helped to improve the performance of the CAD scheme. The optimized parameters restrict the range of the features, which characterize the true signals, allowing the cut of the number of false alarms without losing too many true signals. In addition, we have investigated the influence of the fluctuations in the parameter values on the performance of the CAD system. We have also tested the results of the detection scheme in the presence of cases not used in training.

Another purpose of the present paper was to study the feasibility of using our cluster of workstations to resolve this complex optimization problem instead of moving towards a much more expensive and rarely available Massively Parallel Processing (MPP) system. Moreover, we wanted to obtain the best results possible without disturbing the routine activities of other users. Therefore, we have performed all of our tests during "normal" users' activities who actually do not run long jobs. Thus, we wrote a parallel algorithm which can exploit at the best the independence of the problem data by subdividing the natural individuals data domain, the whole image database in smaller *chunks*. Then, we have created an image-individual pair, which breaks the task of each computer in smaller ones and achieves an excellent workload balancing, by reducing the needs of synchronization at the lowest level. This choice has allowed us to obtain very good results both in terms of quality and efficiency.

Acknowledgments

Images were provided by courtesy of the National Expert and Training Center for Breast Cancer Screening and the Department of Radiology at the University of Nijmegen, the Netherlands.

This work is supported by the Italian National Institute for Nuclear Physics (CALMA project).

Authors would like to thank Prof. F.-L. Navarria for his interest in the present work.

References

1. H. P. Chan, K. Doi, R. A. Schmidt, C. E. Metz, K. L. Lam, T. Ogura, Y. Wu, C. J. Vyborny, and H. MacMahon, *Invest. Radiol.* **25**, 1102 (1990).
2. A. Bazzani, A. Bevilacqua, D. Bollini, R. Brancaccio, R. Campanini, N. Lanconelli, and D. Romani, *Int. J. Mod. Phys. C* **11**, 901 (2000).
3. A. Bazzani, A. Bevilacqua, D. Bollini, R. Campanini, N. Lanconelli, A. Riccardi, and D. Romani, *Proceedings of IWDM 2000*, in press, Toronto, Canada, 2000.
4. K. Jinwoo and B. P. Zeigler, *JPDC* **32**, 90 (1996).
5. L. B. Jack and A. K. Nandi, *Proceedings of ESANN99* (D-facto Publications, Brussels, 1999), p. 313.
6. V. Schnecke and O. Vornberger, *IEEE TEC* **1**, 266 (1997).
7. C. Lucasius and G. Kateman, *TrAC* **10**, 254 (1991).
8. S. Hahn, K. H. Becks, and A. Hemker, *Proceedings AIHEP 92* (World Scientific, Singapore, 1992), p. 255.
9. M. Anastasio, H. Yoshida, R. Nagel, R. M. Nishikawa, and K. Doi, *Med. Phys.* **25**, 1613 (1998).
10. H. Yoshida, M. Anastasio, R. Nagel, R. M. Nishikawa, and K. Doi, *Proceedings of IWCAD 1997* (Elsevier Science B. V., The Netherlands, 1997).
11. T. Ema, K. Doi, R. M. Nishikawa, Y. Jiang, and J. Papaioannou, *Med. Phys.* **22**, 161 (1995).
12. E. Cantú-Paz, Ph.D. Thesis, Univ. of Illinois, Urbana, 1999.
13. E. Cantú-Paz, Report No. 97003, Univ. of Illinois, Urbana, 1997.
14. E. Cantú-Paz, Report No. 97004, Univ. of Illinois, Urbana, 1997.
15. A. Bevilacqua, *Informat.* **23**, 49 (1999).
16. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine — A Users' Guide and Tutorial for Networked Parallel Computing* (MIT Press, London, 1994).