Evolutionary Algorithms in Image Reconstruction from Limited Data*

ANDREI BĂUTU†, ELENA BĂUTU, CONSTANTIN POPA

Abstract. We consider in this paper two classes of evolutionary methods for improving the ART Kaczmarz procedure: genetic and, respectively particle swarm optimization algorithms. They are combined in various ways with the classical Kaczmarz projection method, in two classes of hybrid algorithms. Experiments ilustrating the efficiency of these new methods are presented for consistent least-squares formulation of some simulated reconstruction problems in borehole electromagnetic geotomography.

AMS Subject Classifications: 65F10, 65F20, 68T20

Key words and phrases: linear least-squares problems, Kaczmarz algorithm, computerized tomography, electromagnetic geotomography, genetic algorithms, particle swarm optimization

1 Limitation of data in practice and theory

In this section we shall analyse from both technical and mathematical viewpoints the "data limitation" aspect appearing in two very important practical problems: medical computerized tomography (MCT, for short) and electromagnetic geotomography (EGT, for short). The corresponding idealized and simplified (two dimensional sections) situations are presented in figures 1 and 2 below.

In figure 1, related to the MCT, we supposed that for each position of the CT scanner, only one X-ray is emited (S_iR, $i = 1, 2, \ldots, m$). S_i is the source and R is the receptor; the body-section which is analysed is "contained" in the rectangular region $ABCD$.

*The paper was supported by the PNCDI INFOSOC Grant 131/2004

†Merchant Marine Faculty, "Mircea cel Batrân" Naval Academy, 1^{st} Fulgerului St, Constanta, 900218, Romania; e-mail: abautu@gmb.ro

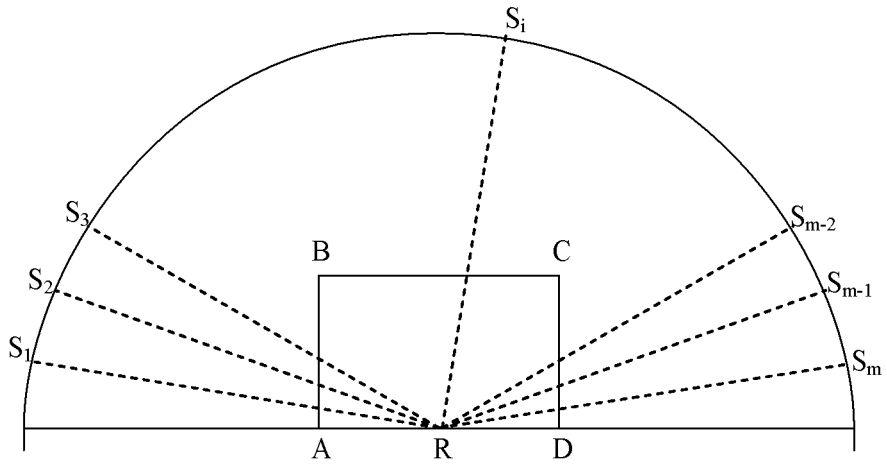


Figure 1: The MCT problem

Figure 2 describes an EGT problem; $ABCD$ is the rectangular earth region which is analysed, AB and CD are holes in which are introduced electromagnetic waves sources $S_1, S_2, \ldots, S_p$ and receptors $R_1, R_2, \ldots, R_q$, respectively (see [6]).

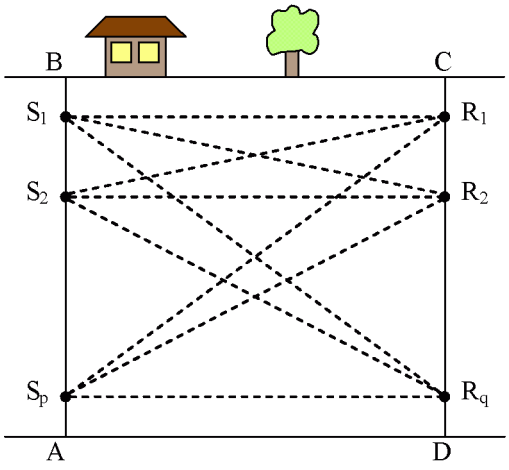


Figure 2: The EGT problem

In both cases, the rectangular regions $ABCD$ are uniformely discretized in a number n of pixels $P_1, P_2, \ldots, P_n$ (see figure 3).

The mathematical model of the reconstruction procedure, for both EGT

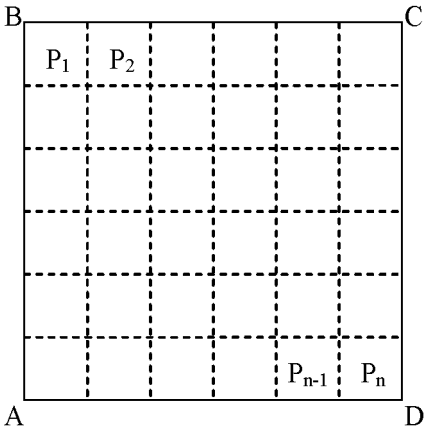


Figure 3: Pixels discretization

and MCT problems is a linear least-squares formulation

$$\|Ax - b\| = \min! \tag{1}$$

with A an $m \times n$ matrix and $b \in I\!R^m$. The right hand side b is constructed by measuring the X-rays or electromagnetic waves intensities at sources and receptors (see [2], [5] for details). Concerning the matrix A, the number n of its columns is exactly the number of pixels in the discretization from figure 3, whereas the number m of its rows corresponds to the number of X-rays in the MCT case (S_iR denoted by E_i, $i = 1, 2, \ldots, m$) or electromagnetic waves source-receptor combinations in the EGT case (S_kR_l, $k = 1, \ldots, p$, $l = 1, \ldots, q$ denoted by E_i, $i = 1, 2, \ldots, m = pq$). The value of the $(A)_{ij}$ component is the length of the segment intersection between the E_i ray and the pixel P_j (see figure 4). If such an intersection is empty, the corresponding $(A)_{ij}$ coefficient is set to 0. Following such a construction, the matrix A becomes sparse, rank-defficient and ill-conditioned (see [2] for details). Moreover, because of measurement errors, the right hand side b fails to belong to the range of A thus, the problem (1) becomes inconsistent. But, for simplifying the presentation, we shall consider in this paper only the consistent case of (1) and let the inconsistent one for a future work. By "data" associated to problem (1) we understand the matrix A and the vector b. Moreover, because the number n of pixels in the discretization from figure 3 is imposed by technical reasons, we may consider our "data" as essentially determined by m, the number of X-rays/electromagnetic waves that are used for scanning

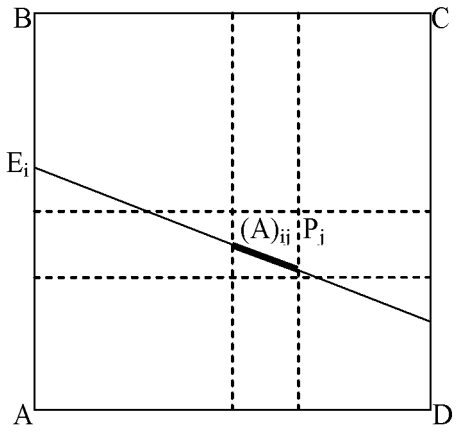


Figure 4: Matrix coefficients

$ABCD$. From a practical point of view, this number is limited at least by the following two reasons:

- for MCT - a too big number m of X-rays used for scanning a body can become (very) dangerous for its health;
- for EGT - in practice, the underground analysed region $ABCD$ is very big, thus the length of the holes AB and CD is so; then, if we would like a "complete" scanning of the area we would need a very big number of sources and receptors, which is not always possible from technical reasons.

Let's now analyse from a mathematical view point the above described "data limitation". In this sense we shall denote by $A^t, N(A), R(A), A^+$ the transpose, null space, range and Moore-Penrose pseudoinverse of A and by $LSS(A;b)$, x_{LS} the set of all least squares solutions of (1) and its (unique) minimal norm one; as previously mentioned we shall suppose that

$$b \in R(A). \tag{2}$$

It is well known (see [1]) that

$$LSS(A;b) = x_{LS} + N(A), \quad x_{LS} \perp N(A) \tag{3}$$

where $\perp$ denotes the orthogonality w.r.t. the euclidean scalar product $\langle \cdot, \cdot \rangle$. Let a_i denote the i^{th} row of A; we know that

$$R(A) = span\{a_1, \ldots, a_m\} \tag{4}$$

and the following orthogonal decomposition holds

$$\mathbb{R}^n = N(A) \oplus R(A^t), \tag{5}$$

together with the corresponding dimensional relation

$$n = \dim(N(A)) + \dim(R(A^t)) \tag{6}$$

According to (3), each solution of (1), $x^* \in LSS(A; b)$ can be expressed as

$$x^* = x_{LS} + P_{N(A)}(x^*), \tag{7}$$

where $P_S(x)$ denotes the (euclidean) orthogonal projection onto the subspace S. Unfortunatelly, both direct and iterative solvers for (1) generally give us only the (unique) solution x_{LS} (see e.g. [1], [2]). If the null space $N(A)$ is "big enough" w.r.t the support space $\mathbb{R}^n$ and x^* has a corresponding "big" component, the difference

$$x^* - x_{LS} = P_{N(A)}(x^*) \tag{8}$$

can become enough important to destroy the accuracy of the reconstructed image. The fact that $N(A)$ is "big enough" is essentially related to its dimension (compared to the dimension of $R(A^t)$); thus, if data are limited ($m < n$) we expect that

$$\dim(R(A^t)) \leq m < n, \tag{9}$$

which according to (6) gives us

$$\dim(N(A)) = n - \dim(R(A^t)) > 0. \tag{10}$$

If data are not limited, i.e.

$$m \geq n \tag{11}$$

we expect a bigger dimension for $R(A^t)$ or even a full rank matrix A, i.e.

$$\text{rank}(A) = \dim(R(A^t)) = n, \tag{12}$$

which would mean that

$$N(A) = \{0\}, \tag{13}$$

thus

$$LSS(A; b) = \{x_{LS}\}. \tag{14}$$

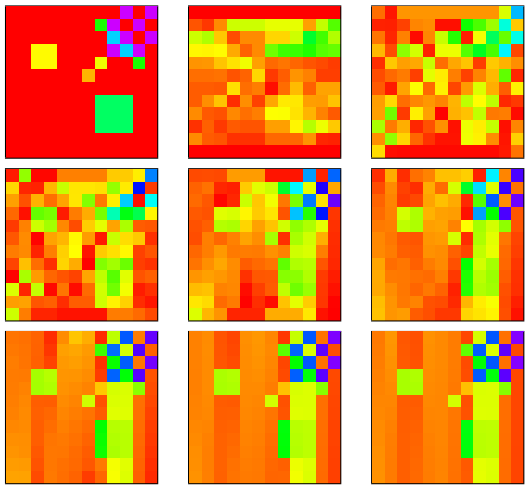


Figure 5: Original image (upper left) and reconstruction results for p sources and p receivers (where $p \in \{6, 8, 10, 12, 16, 24, 36, 48\}$)

Figure 6: Color mapping scale

In these cases, the reconstructed image (which will be much closer or identical to x_{LS}) will also be much closer or identical to the exact one x^* from (7). All the previous theoretical considerations will be ilustrated in the following example in which a real image reconstruction EGT problem is simulated (the same procedure will be used in the experiments from section 3 of the paper).

Example 1 *Our simulation procedure is the following: we consider an image artificially created (see figure 5) as a vector $x^{ex} \in \mathbb{R}^n$, for a given number $n \geq 2$ of pixels (as in figure 4). Each component x_i^{ex} is a real number in the interval $[0, 1]$ and corresponds to the color maping scale from figure 6. This gives us the coloured original image in figure 5 (in this case we have $n = 144$). Then, the original image area was scanned as in figure 2, by using $p \geq 1$ sources and $q \geq 1$ receptors, equally distributed on AB and CD. In this way we obtained the $m \times n$ system matrix A (see figure 4) with $m = pq$.*

The right hand side $b \in \mathbb{R}^m$ was defined by

$$b = Ax^{ex}, \tag{15}$$

Scanning ($p = q$)	m	n	$\text{rank}(A)$	$\dim(N(A))$
6×6	36	144	35	1
8×8	64	144	61	3
10×10	100	144	96	4
12×12	144	144	120	24
16×16	256	144	131	13
24×24	576	144	133	11
36×36	1296	144	133	11
48×48	2304	144	133	11

Table 1: Limited data tests characteristics

such that the problem (1) becomes consistent. For the tests from figure 5 we used $p = q$ *(i.e* $m = p^2$*) with* $p \in \{6, 8, 10, 12, 16, 24, 36, 48\}$ *(for* $p = 6, 8, 10, 12$ *we have* $m \leq n$*, i.e. the limited data case). The values of the* $\text{rank}(A)$ *and* $\dim(A)$ *were computed in each case and are presented in Table 1. The original image* x^{ex} *from figure 5 has components in* $N(A)$*, thus (see (7))*

$$x^{ex} \neq x_{LS}. \tag{16}$$

We then applied to (1) the classical Kaczmarz algorithm (KA, for short) with the initial approximation $x^0 = 0$ *(which gives us* x_{LS} *of (1) as the limit of the sequence of approximations (see e.g. [1], [2], [5]). After 100 iterations for the cases presented in Table 1 we got the results from figure 5. We can observe there that in the "limited data" situations (*$p = q = 6, 8, 10, 12$*) the reconstructed images (which corresponds to* x_{LS}*) are far from the original one (the first 4 images following the original one), whereas for "enough much data" (e.g.* $p = q = 24, 36, 48$*) the reconstructed images are much closer to the original (in all these cases - see Table 1 - the null space of* A *has dimension 11, which is small w.r.t.* $\dim(R(A^t)) = \text{rank}(A) = 133$*).*

Note. *The shadows appearing in the bottom row 3 images in figure 5 are due to the following limitation of our simulation procedure: as mentioned before, the original image* x^{ex} *has components in* $[0, 1]$ *corresponding to the color mapping scale from figure 6, whereas the* x_{LS} *solution vector can have also components outside* $[0, 1]$*. These ones will be outside the color maping scale range and will be automatically associated with the colours appearing in the shadows from figure 5. Work is in progress for overcoming this limitation in our future simulation research.*

Let's now return to the limited data case, in which $\dim(N(A))$ is "big enough" such that $P_{N(A)}(x^*)$ has an important contribution in x^*, beside x_{LS} (see (7)). In order to get a good enough approximation of x^*, e.g. with Kaczmarz's algorithm used in Example 1, we need to start it with an initial approximation $x^0 \in I\!R^n$ for which

$$P_{N(A)}(x^0) \approx P_{N(A)}(x^*). \tag{17}$$

According to (7) we will then have

$$\lim_{h\to\infty} x^h = P_{N(A)}(x^0) + x_{LS} \approx P_{N(A)}(x^*) + x_{LS} = x^*, \tag{18}$$

thus, a much better approximation for x^*. For generating such a "good" initial approximation x^0 as in (17) we decided to use two evolutionary algorithms. They will be described in the next section, whereas in the last one we shall combine one of them with the previous Kaczmarz algorithm in our numerical experiments.

2 Evolutionary Algorithms

We attempted to use two evolutionary algorithms to solve linear least squares problems formulated as (1): a **genetic algorithm** (GA, for short) and a **particle swarm optimization** algorithm (PSO, for short). Evolutionary algorithms maintain a set of possible solutions which they try to improve by evolving them into more performant ones.

Some random factors appear in evolutionary algorithms, which means that they may generate different outcomes on different runs. In order to obtain an overall good performance for solutions these randoms factors are controlled through different methods and parameters.

Each posible solution is rated by a fitness function. Both algorithms presented here use a fitness function that tries to minimize the differences between simulated and measured results. The fitness function is defined as follows:

$$f_f(x) = \frac{1}{1 + \|Ax - b\|} \tag{19}$$

where x is an n-dimensional vector with components in $[0, 1]$, which represents the current image (see Example 1).

2.1 Genetic Algorithm

We attempted to extend the genetic algorithm for unary function optimization described in [4]. The set of possible solutions in a genetic algorithm is called a **population**. The population may have fixed or dynamic size, depending on the implementation. Each solution from the population is called a **chromosome** (or **individual**). Data which define a chromosome state are called **genes**. Most genetic algorithms work by evolving their population with the help of three genetic operators: **mutation, crossover** and **selection**.

In our implementation of a GA, the algorithm maintains a fixed size population of *pop_size* chromosomes. Each chromosome maintains a vector of n double precision floating point numbers from $[0, 1]$. The elements of this vector are the chromosome's genes. The values of the i^{th} gene represents the absortion value of the i^{th} pixel in the image.

Mutation is a unary operator that affects the genes of a single chromosome. It modifies each gene with a given probability by replacing the old value with a randomly generated value from $[0, 1]$. Each gene suffers mutation with *mut_rate* probability.

Crossover is a binary operator which combines a pair of chromosomes (called **parents**) to obtain a pair of new chromosomes (called **offsprings**) with features from each parent. Chromosomes are selected for mating with *cross_rate* probability. For each pair of chromosomes, a random crossover point is selected and genes to the left of that point are swapped between chromosomes. In our implementation, offsprings immediatly replace their parents in the population. This is not a mandatory feature of genetic algorithms, but it provides greater performance.

Selection is a population-wide operator. Essentially, a selection operator should create a new population based on the old one. There are many possible selection operators available, but we decided to implement a classical Monte Carlo selection scheme (called **roulette wheel selection**). Monte Carlo selection copies chromosomes from the old population into the new one with probabilities related to their fitness values. Thus a better fit chromosome will have more chances to have copies of itself in the next population than a poor fit one.

For our GA the values range of the above described parameters are the following: for *pop_size* from 10 to 200, *mut_rate* - 1% to 25%, *cross_rate* - 30% to 70%, depending on the algorithm operators choice and problem class.

2.2 Particle Swarm Optimization Algorithm

Next, we attempt to create a n-dimensional extension of the PSO algorithm described in [3]. The set of possible solutions in a particle swarm algorithm is called a **swarm**. The swarm may have fixed or dynamic size, depending on the implementation. Each solution from the swarm is called a **particle**. A particle state is defined by **current position, velocity, memory** (or **history**) and **neighbours**. Most PSO algorithms work by evolving their swarm by adjusting the speed, position and memory of each particle.

In our implementation, each particle encodes its state as a vector of tuples of double precision floating-point numbers from $[0, 1]$. The values in the i^{th} tuple represent the current position, velocity and best so far position of the particle in i^{th} search dimension. The position component of the i^{th} tuple represents the absortion value of the i^{th} pixel in the image, and decodes into the color of that pixel.

In the algorithm we have a fixed-size set of *part_count* particles. Each particle evolves based on its history (best so far position) and its neighbours (only one neighbour in our implementation). On each iteration, velocity and position components of particles are updated with the following formulas:

$$v'_{t+1} = v_t \cdot inertia + rand() \cdot cognitive \cdot (b_t - p_t) + rand() \cdot social \cdot (n_t - p_t) \quad (20)$$

$$v_{t+1} = min(v_{max}, max(-v_{max}, v'_{t+1})) \quad (21)$$

$$p'_{t+1} = p_t + v_{t+1} \quad (22)$$

$$p_{t+1} = min(1, max(0, p'_{t+1})) \quad (23)$$

For our PSO algorithm the values range of the above described parameters are the following: for *part_count* from 10 to 100, *inertia* - 1% to 50%, *cognitive* and *social* - 75% to 200%, v_{max} - 0.1 to 1.0, depending on problem class.

2.3 Hybrid Algorithms

As we presented in the experiments from section 3 of the paper, the genetic algorithm performed quite bad compared to PSO and Kaczmarz ones. So, we decided to create hybrid algorithms only from Kaczmarz and PSO.

First hybrid algorithm (FHA, for short) has two stages. During the first stage it runs the PSO algorithm described earlier. In the second stages it uses the best solution from the first stage as the starting approximation for Kaczmarz algorithm.

Second hybrid algorithm (SHA, for short) runs Kaczmarz and PSO algorithms in an alternate manner, by applying after each iteration of PSO (on each particle) one step of the Kaczmarz algorithm.

3 Experiments

We present our results for 4 image reconstruction experiments. The simulation procedure is the same as in Example 1. The 4 original images are showed in figure 7 and their characteristics in Table 2 (left to right, according to figure 7). For each image, we ran all five algorithms with various settings, but we present results only for the 10th and 100th iteration for the following ones:

- KA: no settings required;
- GA: $pop\_size = 50$, $mut\_rate = 5\%$, and $cross\_rate = 70\%$;
- PSO: $part\_count = 50$, $vmax = 1.0$, $inertia = 0.3$, $cognitive = 1.2$, and $social = 1.2$;
- FHA and SHA: same settings as for PSO.

Image	Size (pixels)	Source	Unique colors
Test 1	8×8	Drawing	9
Test 2	12×12	Drawing	8
Test 3	16×16	Drawing	11
Test 4	20×20	Scanned photo	71

Table 2: Test images properties

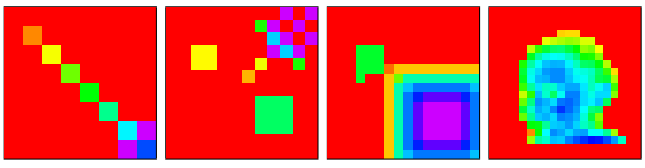


Figure 7: Test images

3.1 After 10 iterations

As we may see in figures 8 and 9, after 10 iterations GA and PSO failed to produce a "good" reconstruction even for the simple Test 1 image; GA's image is useless, but PSO's image began to shape. We consider this to be a normal behaviour since 10 iterations are not enough to allow the algorithms to establish a good search direction.

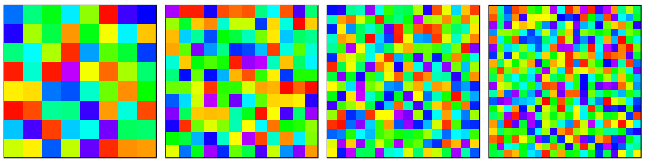


Figure 8: GA results after 10 iterations

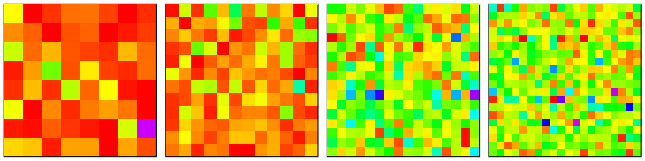


Figure 9: PSO results after 10 iterations

As expected, KA found some shapes (see figure 10), but images are still affected by noise, which is common in (classical) Kaczmarz image reconstructions (see [2], [6]).

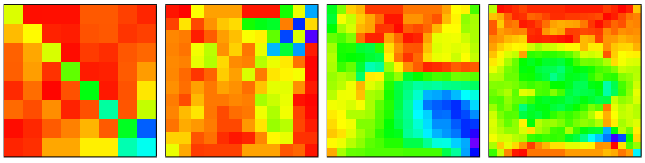


Figure 10: KA results after 10 iterations

Powered by KA, FHA found some shapes (see figure 11), too. However, because of the "bad" starting approximation generated with PSO during the first stage of FHA, reconstructed images are not as good as those of KA.

Results of SHA are surprisingly good (see figure 12). Test 1 image is almost perfect and other images have very well defined shapes. It seems that

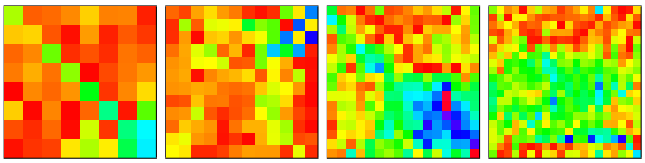


Figure 11: FHA results after 10 iterations

KA drives the reconstruction process towards the "good" image, while PSO helps filtering the image in order to eliminate the noise.

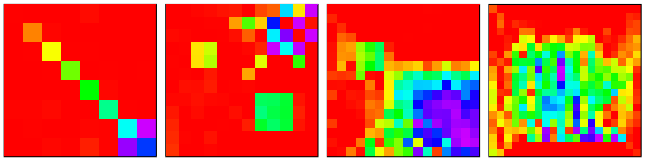


Figure 12: SHA results after 10 iterations

3.2 After 100 iterations

After 100 iterations, GA still doesn't produce a good image (figure 13). Maybe a different encoding or operators should be tried (as we said, there are many genetic operators and they can be combined to form many algorithms).

PSO has improved its Test 1 and Test 2 results (see figure 14), but results are not very satisfactory and it has no valid solution for Test 3 and Test 4 images.

KA has improved very little its results for Test 1 and Test 2, too, but the noise of the computations is making further improvements difficult (see figure 15). For Test 3 and Test 4 it has no satisfactory results.

FHA performed much better than PSO and a little better than KA (see figure 12). Reconstructed images are affected by noise of the KA in the second stage of the algorithm.

SHA improved its Test 2 result to almost perfect and found very good ones for Test 3 and Test 4 (see figure 17). All its results are much better than those offered by Kaczmarz algorithm alone or the other evolutionary algorithms presented in this article.

Figures 8-17 show the reconstruction result as the original "square" images in figure 7. These visual comparisons give us information on the "visual

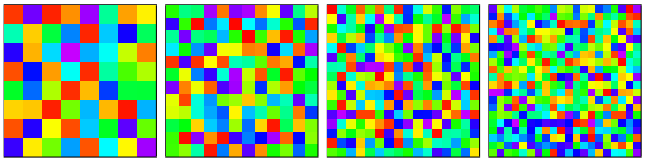


Figure 13: GA results after 100 iterations

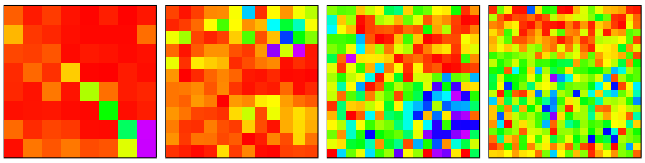


Figure 14: PSO results after 100 iterations

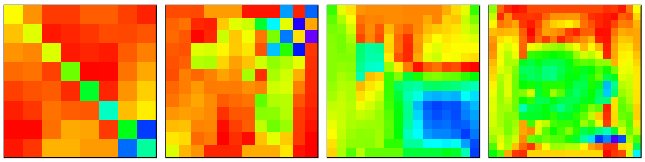


Figure 15: KA results after 100 iterations

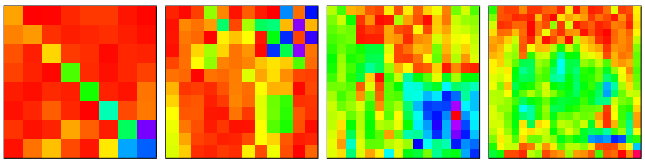


Figure 16: FHA results after 100 iterations

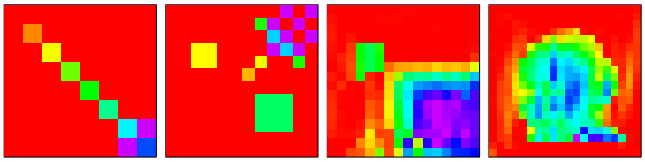
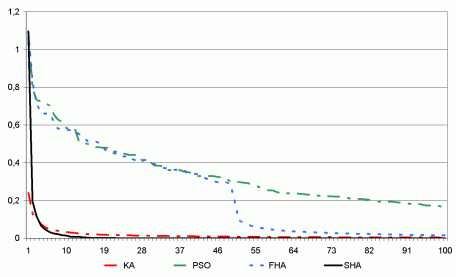
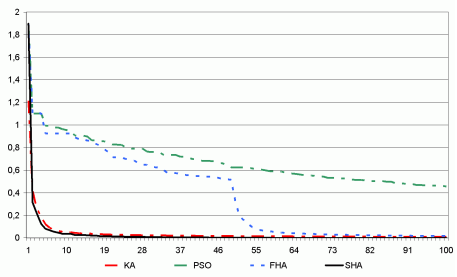


Figure 17: SHA results after 100 iterations

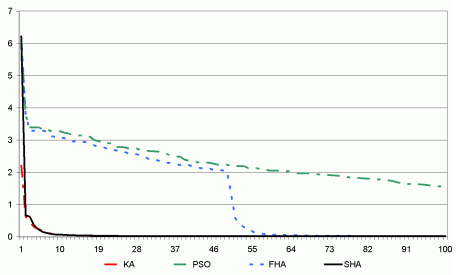
part" of the approximation error. According to the convergence properties of the 4 used algorithms, charts in figure 18 and 19 show the evolutions of residual ($\| Ax^k - b \|$) and absolute error euclidean norm, respectively ($\| x^{ex} - x^k \|$), during 100 iterations for KA, PSO, FHA and SHA. We decided not to present similar charts for GA since the results are of no interest (e.g. the residual norm in the GA was almost constant).



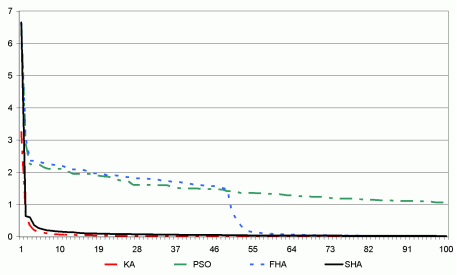
Test 1 over 100 iterations



Test 2 over 100 iterations



Test 3 over 100 iterations



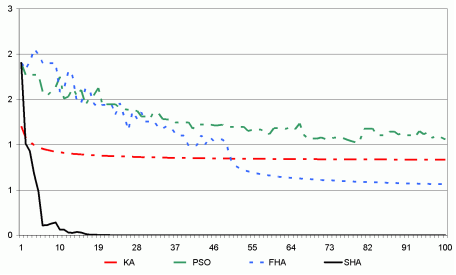
Test 4 over 100 iterations

Figure 18: Residual norm variations

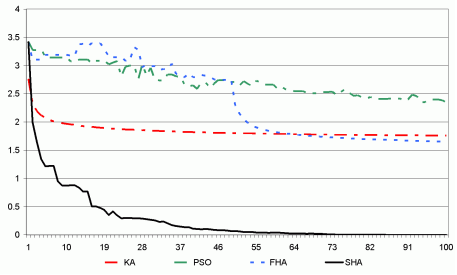
Final comments. The consideration and results from this paper are at a very begining. The first next step will be to apply them to inconsistent least-squares formulations of the type (1), whereas a further research will be concerned with a theoretical analysis of the new obtained image reconstruction methods, together with comparisons with other new and efficient techniques in the field (see e.g. [2]).

References

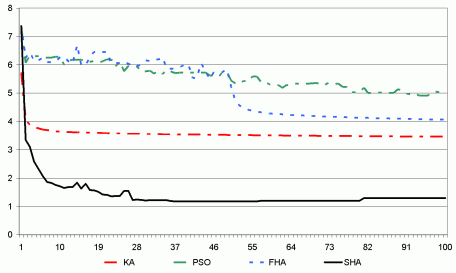
[1] Björck, A., *Numerical methods for least squares problems*, SIAM Philadelphia, 1996.



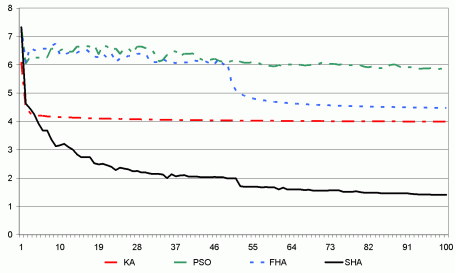
Test 1 over 100 iterations



Test 2 over 100 iterations



Test 3 over 100 iterations



Test 4 over 100 iterations

Figure 19: Distance value variations

[2] Censor Y., Stavros A. Z. *Parallel optimization: theory, algorithms and applications*, "Numer. Math. and Sci. Comp." Series, Oxford Univ. Press, New York, 1997.

[3] Kennedy, J., Eberhart, R., *Particle Swarm Optimisation*, Proceedings of IEEE International Conference on Neural Networks, Perth, Western Australia, (1995), Vol. 3, 1942-1948.

[4] Michalewicz, Z., *Genetics Algorithms + Data Structures = Evolution Programs*, Springer - Verlag, New York, 1996.

[5] Natterer F., *The Mathematics of Computerized Tomography*, John Wiley and Sons, New York, 1986.

[6] Popa C., Zdunek R., *Kaczmarz extended algorithm for tomographic image reconstruction from limited-data*; Math. and Computers in Simulation, **65**(2004), 579-598.