# Sparse Matrices Reordering using Evolutionary Algorithms: A Seeded Approach

**David Greiner*, Gustavo Montero and Gabriel Winter**

Institute of Intelligent Systems & Numerical Applications in Engineering (IUSIANI)
University of Las Palmas de Gran Canaria. Edificio Central del Parque Científico y Tecnológico, 2ª planta. Campus Universitario de Tafira. 35017 Las Palmas de GC, Spain.
E-mail: dgreiner@iusiani.ulpgc.es, gustavo@dma.ulpgc.es, gabw@step.es

**Key Words**: Evolutionary Algorithms, Sparse Matrices, Reordering

**ABSTRACT**

In this work, it is introduced a methodology for solving the problem of sparse matrices reordering using evolutionary algorithms, which can be handled as a combinatorial NP-class problem. Evolutionary algorithms are more flexible techniques that allow this reordering considering location and also values of the non zero entries of the matrix. Different fitness functions are proposed and studied comparatively. Moreover, the obtained results are compared with a classical procedure, the inverse Cuthill-McKee ordering. Finally, a seeded approach that combines both strategies, whose results outperform the previous ones, is introduced

## 1 INTRODUCTION

Sparse matrices are those matrices which have the majority of their entries as zeros, being the non-zero values a minimum proportion of the total. Their use is frequent and well spread in many fields of science and engineering. As an example, discrete modelling with finite elements or finite volumes in engineering problems follows to a linear system of equations $Ax=b$, frequently governed by a sparse matrix. The resolution of this system can be handled with direct (Gauss, LU-factorization, etc.) or iterative (conjugate gradient, generalized minimal residual GMRES, biconjugate gradient stabilized Bi-CGSTAB, etc.) methods, depending on the problem characteristics. Either the case, the reordering of the entry positions often provides advantages. In direct methods, when the matrix bandwidth is reduced via reordering, the 'fill in' effect (zero entries filling before factoring procedure) is diminished. In iterative methods, the effect of certain preconditioners can be improved via reordering [4,5,18]. This reordering problem belongs to the nondeterministic polynomial time NP-class combinatorial problems.

Classical methods have been developed for this purpose, see e.g. [17]. However, in recent years, bio-inspired and heuristic methods have been proposed, such as tabu search [20] or simulated annealing [16]. The sparse matrix reordering problem has been faced previously also as a graph partitioning problem. In this way, classical methods have been proposed, such as [12], or [6] for parallel computers; but also evolutionary methods, such as [14] in single objective optimization, or [3] considering multiobjective optimization.

Here we introduce a methodology with evolutionary algorithms for sparse matrix reordering, describing the genetic operators, the codification and various fitness functions proposed for a suitable performance. Results are compared with a classical method: the inverse Cuthill-McKee reordering [7,8]. Moreover, a seeded approach is exposed, being its results capable of outperform the previous ones.

The organization of this paper is described as follows: First, the sparse matrix reordering problem is handled in Section 2, describing our chromosome codification, some evolutionary considerations, and the considered fitness functions. Section 3 describes the experimental results, focusing in the evolutionary and seeded approaches, and their discussion. Finally the paper ends with the conclusions section.

## 2 SPARSE MATRICES REORDERING PROBLEM

The resolution of the sparse matrices reordering problem using evolutionary algorithms requires a suitable chromosome codification and the appropriate definition of the fitness function. Both aspects are described in the following subsections, as well as some evolutionary considerations.

### 2.1 CODIFICATION OF THE CHROMOSOME

The variable information among candidate solutions (what differentiates one solution from another) is the only knowledge required to be coded. In addition, this information should be structured in such a way that the information interchange between solutions using crossover or its modification using mutation does not produce an infeasible solution.

The used matrix storage is a compact storage, based in the compressed schema of Radicati [21], where the dynamic memory assignment characteristics of C++ language are considered. Using a compact storage in a sparse matrix is advantageous due to the fact, that only a few number of its values are non-zeros. The sparse matrix (A) is defined through two matrices called position matrix (PA) and value matrix (VA). The position matrix PA stores by rows the position of the non-zero values of each file of the matrix; except the first value, that corresponds to the whole number of non-zero values contained in the PA row. The value matrix VA contains the matrix values, storing in the first value of each row the main diagonal value, and the rest of the values correspond to the non-zero values whose positions are defined in the position matrix.

Each candidate solution (one chromosome) is representative of one defined ordering corresponding to the matrix A. A naïve codification of the chromosome could be a compact matrix with a defined ordering. But considering that only the variable or different information among solutions is required to be coded in the chromosome, it could be limited exclusively to a vector with so many elements as the dimension of sparse matrix A, and defining the ordering of its nodes. So, each chromosome is constituted by the natural numbers from 1 to n, being n the dimension of matrix A, and in a random ordering. This ordering defines the permutation of rows of position and value matrices (PA and VA), as well as the interior permutations of the position values of each row, in such a way that allows to define the new matrix in compact storage (also the values of the independent term vector should be permuted in the same ordering, in order that the solution of the original lineal system would not be altered). Considering an original consecutive ordering of the matrix from 1 to n, then with a sparse matrix dimension of four, an example of matrix reordering is represented in figure 1. It includes PA and VA matrices and chromosome coding.

$$Chromosome\,A=\{1,2,3,4\}\ A=\begin{bmatrix}1 & 0 & -1 & 0\\ 0 & 2 & 0 & 2\\ 1 & 0 & 3 & 0\\ 0 & -1 & 0 & 4\end{bmatrix};\quad PA=\begin{bmatrix}2 & 3\\ 2 & 4\\ 2 & 1\\ 2 & 2\end{bmatrix};\quad VA=\begin{bmatrix}1 & -1\\ 2 & 2\\ 3 & 1\\ 4 & -1\end{bmatrix}$$

$$Chromosome\,A'=\{4,1,3,2\}\ A'=\begin{bmatrix}2 & 2 & 0 & 0\\ -1 & 4 & 0 & 0\\ 0 & 0 & 3 & 1\\ 0 & 0 & -1 & 1\end{bmatrix};\quad PA'=\begin{bmatrix}2 & 2\\ 2 & 1\\ 2 & 4\\ 2 & 3\end{bmatrix};\quad VA'=\begin{bmatrix}2 & 2\\ 4 & -1\\ 3 & 1\\ 1 & -1\end{bmatrix}$$

**Fig. 1.** Original A, PA, VA, reordered matrices A', PA', VA' and their corresponding chromosome codifications

Using this codification, the chromosome treatment is analogous to that corresponding in well known evolutionary problems, such as simple scheduling [13], or the classical travelling salesman problem [15], where a set of cities or control points should be covered in the shortest possible path, and the chromosome can be expressed as the ordering of the cities through the route.

### 2.2 EVOLUTIONARY CONSIDERATIONS

The used crossover and mutation operators are position-based crossover and order-based mutation [19].

$$Candidate\,Solution\,1=\{1,2,3,4\}\ \ Candidate Solution\,2=\{4,3,2,1\}$$

$$Crossover\,Mask=\{0,1,0,1\}$$

$$New\,Candidate\,Solution\,1=\{2,3,4,1\}\ New\,Candidate\,Solution\,2=\{3,2,1,4\}$$

**Fig. 2.** Crossover example

The position-based crossover consists in fixing by a uniform and random crossover mask, those positions that will be interchanged by two parents. To maintain the coherency in the chromosome representation, the remaining positions are maintained in the original relative ordering in the chromosome, and they are filled in the remaining gaps. An example of the procedure is showed in figure 2 for matrix A, where a 1 in the mask means crossover of the position and a 0 means no crossover of the position.

The order-based mutation acts by a mutation mask which contains the positions to mutate. It is considered a uniform mask, where every position has the same probability to be altered. Each mutated value interchanges its position with other randomly determined gene. An example of the procedure is showed in figure 3 for matrix A, where the randomly mutated position described by the mask is the fourth.

$$Chromosome=\{2,3,4,1\}\ Mutation\,Mask=\{0,1,0,0\}\ New\,Solution=\{2,1,4,3\}$$

**Fig. 3.** Mutation example

The description of the evolutionary algorithm used is as follows. It is used a steady-state genetic algorithm with duplicates elimination, where two solutions are substituted each generation, deleting in the insertion the two worst solutions. The selection is performed through stochastic universal sampling (SUS) using a ranking ordering for the probabilities definition. A population of 50 individuals is considered, being the crossover rate of 100% and the stop criterion a total of $6{\cdot}10^6$ fitness function

evaluations. The selected mutation rate is of 0.4%, which gives the best results from a set of rates tested among 0.2%, 0.4%, 0.8% and 1.5% in the presented test case.

### 2.3 ABOUT THE FITNESS FUNCTION

Various fitness functions have been considered here for minimizing when solving the sparse matrix reordering problem. Each of them constitutes the resolution of a different reordering problem, but (excluding the fifth fitness function) with the same aim of minimizing the semi-bandwidth of the matrix. They are compared in the results section, where it is shown how its election conditions the final obtained solution. The proposed fitness functions are:

**Fitness Function 1.** The first fitness function considered is the sparse matrix semi-bandwidth: $F1 = max\ |i - j|,\ a_{ij} \neq 0$ (the semi-bandwidth definition is the maximum number of intermediate positions that separate the further term of a row from the principal diagonal of the matrix).

**Fitness Function 2.** The second fitness function considered corresponds with the total size of the whole sparse matrix envelope: $F2 = \sum_i (max\ j - min\ j),\ a_{ij} \neq 0$ (the envelope definition is the total number of matrix positions which is covered among its non-zero entries by rows or by columns, being in symmetric matrix profiles both values equivalents).

**Fitness Function 3.** The third fitness function is evaluated as follows: For each non-zero value of the matrix, the number of positions that is separated from the principal diagonal is calculated. This value is raised to some power, in order to discriminate more favourably those terms that are further from the principal diagonal of the matrix. The whole sum of this evaluation represents the value of the fitness function for a particular ordering of the sparse matrix: $F3 = \sum_i \sum_j |i - j|^{1.4},\ a_{ij} \neq 0$. Here each term is raised to the power of 1.4.

**Fitness Function 4.** This fourth fitness function is a little variation of the previous one, where each term is raised to the power of 3, increasing the discrimination effect: $F4 = \sum_i \sum_j |i - j|^3,\ a_{ij} \neq 0$.

**Fitness Function 5.** For each non-zero value of the matrix, the number of positions that is separated from the principal diagonal is calculated, and this distance is, in addition, multiplied by the own value of the term. The whole sum of this evaluation represents the value of the fitness function for a particular reordering of the sparse matrix: $F5 = \sum_i \sum_j a_{ij}|i - j|,\ a_{ij} \neq 0$. This fifth fitness function is focused mainly to the iterative resolution methods, more concisely to increase the efficiency of the preconditioners in iterative solvers, where the concentration of the terms of higher value around the principal diagonal can favour the convergence of the resolution. The possibility of including the own value of each term in the matrix reordering procedure is a tool that evolutionary methods provide, but is not considered in the classical ones.

**Fitness Function 6.** This sixth fitness function is a little variation of the previous one, where the term corresponding to the position separation is raised to the power of two: $F6 = \sum_i \sum_j a_{ij}|i - j|^2,\ a_{ij} \neq 0$, increasing the importance of the distance to the principal diagonal.

## 3 RESULTS AND DISCUSSION

### 3.1 THE EVOLUTIONARY APPROACH

The test case considered is a sparse matrix of dimension 441, belonging to a convection-diffusion problem handled by finite elements. Their non-zero values are represented as points in left side of figure 4 (x-axis and y-axis represent the row and column positions respectively, and a point shows a non-zero value). This original matrix is included in the initial population in every case.

Final results are compared with the inverse Cuthill-McKee reordering, which is graphically shown in right side of figure 4. The results numerical expressions are shown in table 1, where each reordering solution has the value of every considered fitness function. The values that have guided each solution are represented in bold type. These reorderings are graphically represented in figure 5.

As it can be inferred from the obtained results, considering the semi-bandwidth as fitness function (first considered) does not seem to be an appropriate election for obtaining matrices with reduced profile. The semi-bandwidth is reduced from 428 to 328, but is very far from the value obtained with the inverse Cuthill-McKee ordering. This disappointing behaviour is understandable considering that the semi-bandwidth as fitness function is not a progressive indicator. It covers many different solutions without discriminating better and worse solutions taking into account the number of entries that are nearer the main diagonal. A fitness function that allows more progression among solutions is required. In this direction were suggested the other proposed fitness functions.

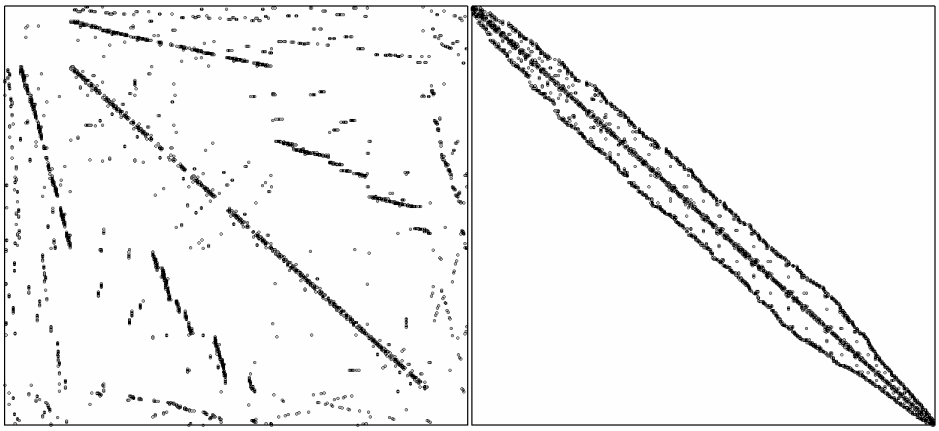

**Fig. 4.** Matrix profiles belonging to the original matrix (left), and the inverse Cuthill-McKee ordering (right)

**Table 1.** Fitness Function (FF) Values of sparse matrix reorderings corresponding to the evolutionary approach

| | Original Matrix | Inv.Cut-McKee M | FF 1 Matrix | FF 2 Matrix | FF 3 Matrix | FF 4 Matrix | FF 5 Matrix | FF 6 Matrix |
|---|---|---|---|---|---|---|---|---|
| FF1 | 428 | **39** | **328** | 175 | 92 | 50 | 432 | 308 |
| FF2 | 109834 | **20768** | 127398 | **21180** | 19011 | 18734 | 36226 | 35727 |
| FF3 | 2629651 | **147468** | 2616623 | 185188 | **128981** | 115656 | 541488 | 450726 |
| FF4 | 16767-E6 | **32.5-E6** | 14250-E6 | 219.6-E6 | 38.6-E6 | **18.6-E6** | 3170-E6 | 2006-E6 |
| FF5 | 12189628 | **1398169** | 14.6-E6 | 1481249 | 1297785 | 1332685 | **1011603** | 921926 |
| FF6 | 1012-E21 | **1002-E21** | 1008-E21 | 990-E21 | 986-E21 | 1004-E21 | 1002-E21 | **991-E21** |

Considering the envelope as fitness function (second considered) improves the obtained results, reducing the semi-bandwidth to 175. The behaviour of the evolutionary algorithm is clearly favoured by the higher progressiveness of this fitness function compared with the first one. However, the envelope criterion is subjected to irregularities that allow a reduction of semi-bandwidth in certain zones of the profile.

The third and fourth proposed fitness functions follow this aim of reduction of irregularities without losing the progressiveness among solutions. Both increase the penalization to the further terms and it is reflected in the semi-bandwidth obtained values (92 and 50), that are improved respect to the first and second fitness functions. The higher grade of the power in the fourth fitness function increases this effect, permitting to achieve a lower final value.

The importance of an adequate election of the fitness function is here evidenced: not always the most obvious proposal (here the semi-bandwidth) is the better. The capability of being a progressive fitness function, that allows identifying slow changes of candidate solutions in the right direction, is in this sparse matrix reordering problem fundamental to achieve success, as results have proven. This characteristic helps the evolutionary algorithm to evolve and learn during its development.

### 3.2 THE SEEDED APPROACH

The evolutionary approach for solving the sparse matrix reordering problem has demonstrated a general good performance in terms of solutions quality, except in the case of the semi-bandwidth as fitness function. In the rest of cases, the values of the evolutionary orderings are lower than those provided by the inverse Cuthill-McKee solution (with the exception of the second fitness function, which is slightly higher). However, improving the results of this evolutionary approach is sought.

**Table 2.** Fitness Function (FF) Values of sparse matrix reorderings corresponding to the seeded approach

| | Original Matrix | Inv.Cut-McKee M | FF 1 Matrix | FF 2 Matrix | FF 3 Matrix | FF 4 Matrix | FF 5 Matrix | FF 6 Matrix |
|---|---|---|---|---|---|---|---|---|
| FF1 | 428 | **39** | **37** | 76 | 65 | 45 | 375 | 269 |
| FF2 | 109834 | **20768** | 20878 | **16899** | 16833 | 18016 | 20938 | 20635 |
| FF3 | 2629651 | **147468** | 148017 | 115334 | **104323** | 110130 | 176032 | 176486 |
| FF4 | 16767-E6 | **32.5-E6** | 32.6-E6 | 34.1-E6 | 20.2-E6 | **16.6-E6** | 341-E6 | 291-E6 |
| FF5 | 12189628 | **1398169** | 1417910 | 1088660 | 1106000 | 1180194 | **956376** | 870037 |
| FF6 | 1012-E21 | **1002-E21** | 1001-E21 | 1001-E21 | 1001-E21 | 1001-E21 | 1001-E21 | **999E-21** |

The inclusion in the initial population of high quality solutions has been proved to be capable of obtaining improved results in terms of convergence speed and/or final quality, both in single and multi-objective evolutionary optimization, see for example [2,9,10,23]. Here is analysed the case of inserting the inverse Cuthill-McKee sparse matrix ordering as a high quality solution in the initial population, what is called seeded approach.

In figure 6 the best of each fitness functions matrix reorderings including this solution in the initial population, are also shown. Their numerical expressions are shown in table 2, where each reordering has the value of every considered fitness function. The values that have guided each solution are represented in bold type.

As it can be inferred from the obtained results of this seeded approach, in general the values are improved compared with the evolutionary one, even in the case of the first

fitness function (semi-bandwidth). This seeded approach allows for obtaining better sparse matrix reorderings in terms of quality compared with the sole classic Cuthill-McKee method.

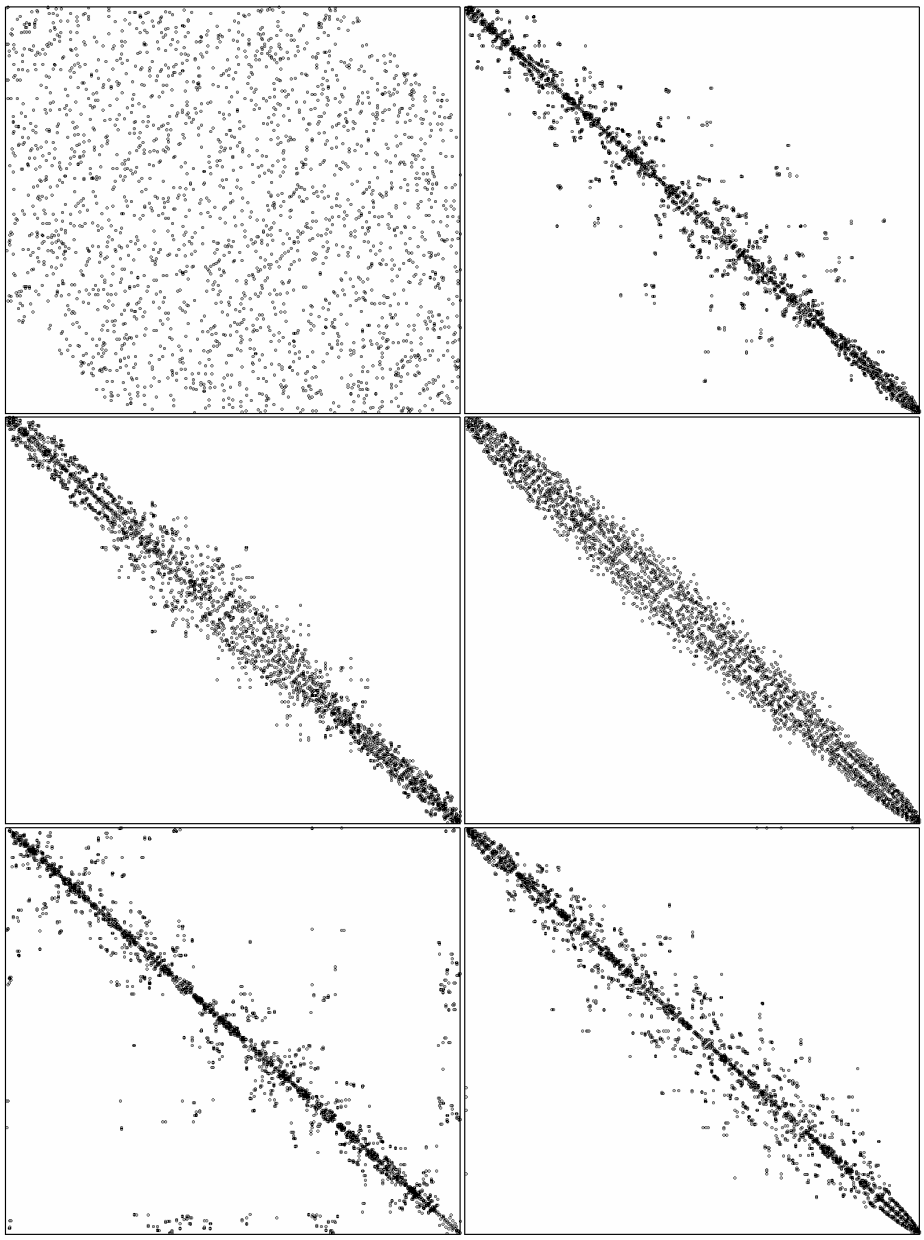

**Fig. 5.** Matrix profiles belonging to the fitness function 1 to 6 best reorderings results, for evolutionary approach

### 3.3 DISCUSSION

From tables 1 and 2 can be inferred that sparse matrices reorderings obtained from functions 5 and 6 introduce a rise in the values of their functions 1 to 4. It can be observed also in figures 5 and 6, where the two inferior graphs in both figures represent this reorderings: some points are further from the principal diagonal than in previous pictures -this effect is reduced in function 6 with respect to function 5, because of its raised power position-term-. This occurs because the own value of the non-zero entry in the fitness function has been taken into account. In the case that this value is sufficiently low, its contribution to the fitness function summation is small, even in the case of

being more separated to the main diagonal of the matrix. This consideration of the own value is not possible in the classic reordering procedures, and gives a possible potential competitive advantage to evolutionary algorithms reordering methods.

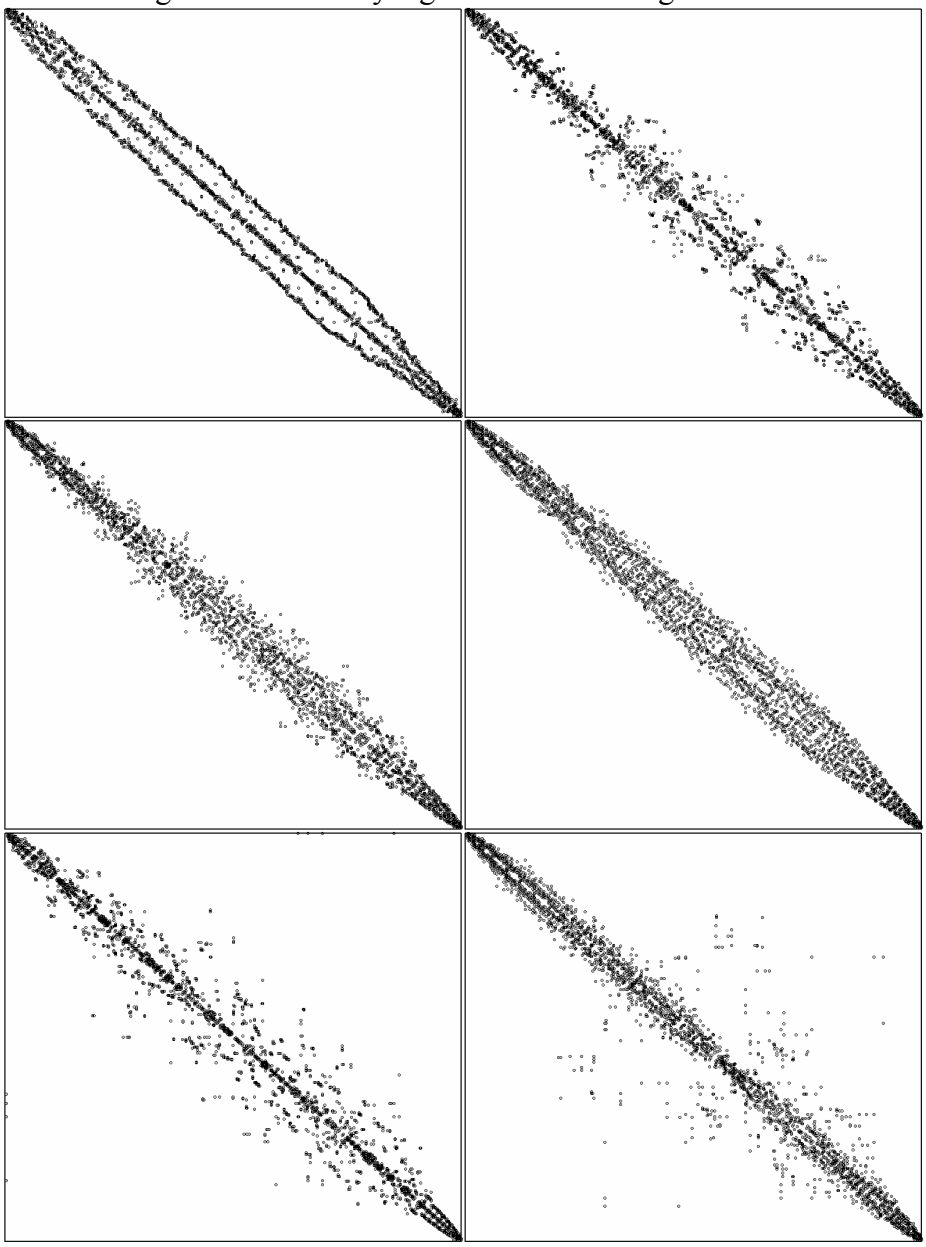

**Fig. 6.** Matrix profiles belonging to the fitness function 1 to 6 best reorderings results, for seeded approach

We have solved the system with the BiCGSTAB method [22], using two different preconditioners: Jacobi and SSOR [1]. The number of iterations corresponding to the different sparse matrix reorderings are shown in table 3 for a stop criterion based in norm 2. It is expressed in equation 1,

$$\frac{\|b - Ax_i\|_2}{\|b\|_2} < 10^{-10} \qquad (1)$$

being $Ax=b$ the linear system.

**Table 3.** Number of BiCGSTAB Solver Iterations required by the sparse matrices reorderings with the Jacobi and SSOR preconditioners

| | | Original Matrix | Inv. CutMK Matrix | FF 1 Matrix | FF 2 Matrix | FF 3 Matrix | FF 4 Matrix | FF 5 Matrix | FF 6 Matrix |
|---|---|---|---|---|---|---|---|---|---|
| Evolutionary Approach | Jacobi | 106 | --- | 101 | 105 | 106 | 104 | 106 | 103 |
| | SSOR | 42 | --- | 56 | 69 | 75 | 77 | 74 | 66 |
| Seeded Approach | Jacobi | --- | 83 | 84 | 84 | 84 | 89 | 83 | 86 |
| | SSOR | --- | 28 | 27 | 30 | 30 | 30 | 30 | 33 |

Due to the relative low order (441) of this matrix test case, the influence of the ordering in the iterations or resolution time is low. In this case, the computational cost of the reordering with evolutionary algorithms, is much higher than the cost of the system resolution by the Krylov method. Nevertheless, in systems of greater order, where their resolution cost is more expensive, it is expected that this reordering cost is compensated, especially when the system has to be solved many times. The advantages of the reordering influence when solving a linear system of equations is increased with the rise of the system size. This fact also increases the complexity of the combinatorial problem of reordering, and consequently, the calculation time. This balance has to be further analyzed in matrices of greater order or complexity.

## 4 CONCLUSIONS

This paper presents a successful methodology for sparse matrix reordering using evolutionary algorithms. Particularly, the introduced seeded approach -that consists in including the inverse Cuthill-McKee reordering in the initial population of the evolutionary algorithm- has proven to outperform in terms of quality of the solution the classical one. Even more, evolutionary algorithms allow the consideration of the value of the terms in the ordering procedure, which could be of particular interest for the preconditioning of iterative solvers.

The advantages that an improved reordering can provide are especially outstanding in the case of performing design optimization with evolutionary algorithms, where many resolutions of matrix systems are required (e.g. finite or volume elements); but only one reordering may be necessary and applicable to the whole process. In that sense, further work should be advisable for achieving lower calculation times in the seeded approach procedure, including suitable operators and parameters of the evolutionary part.

## REFERENCES


[1] Axelsson O., “Iterative Solution Methods”, Cambridge University Press, (1994).

[2] Balling R., “The Maximin Fitness Function: Multiobjective City and Regional Planning”, Evolutionary Multi-Criterion Optimization, Lect. Notes in Comp. Scien. 2632 (2003) 1-15.

[3] Banos R., Gil C., Montoya M.G., Ortega J., “A new pareto-based algorithm for multi-objective graph partitioning”, Computer and Information Sciences - ISCIS, Lecture Notes in Computer Science 3280 (2004) 779-788.

[4] Benzi M., “Preconditioning techniques for large linear systems: A survey”, Journal of Computational Physics, 182; 2 (2002) 418-477.

[5] Benzi M., Szyld DB., van Duin A. “Orderings for incomplete factorization preconditioning

of nonsymmetric problems", SIAM J Sci Comput 20, 5, (1999),1652-1670.

[6] Camarda K.V., Stadtherr M.A., "Matrix ordering strategies for process engineering: graph partitioning algorithms for parallel computation", Computers and Chemical Engineering 23, (1999) 1063-1073.

[7] Cuthill E., "Several strategies for reducing the bandwidth of matrices", In: Sparse Matrices and Their Applications, p. 157, edited by D. J. Rose, R. A. Willoughby (New York, 1972).

[8] Cuthill E., McKee, "Reducing the bandwidth of sparse symmetric matrices", Proceedings ACM National Conference, New York (1969) 157-172.

[9] Greiner D., Emperador JM., Winter G., "Single and Multiobjective Frame Optimization by Evolutionary Algorithms and the Auto-adaptive Rebirth Operator", *in Computer Methods in Applied Mechanics and Engineering*, Elsevier, 193 (2004) 3711-3743

[10] Greiner D., Emperador JM., Winter G., Multiobjective optimization of bar structures by Pareto GA, in: *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2000)*, Barcelona, CIMNE, 2000.

[11] Greiner D., Montero G., Winter G., "Reordering Sparse Matrices with Evolutionary Algorithms", VIII National Congress of Applied Mathematics (2003), Tarragona, Spain.

[12] Gupta A., "Fast and effective algorithms for graph partitioning and sparse-matrix ordering", IBM Journal of Research and Development, 41; 1-2 (1997) 171-183.

[13] Hart E., Ross P., Corne D., "Evolutionary Scheduling: A Review", Genetic Programming and Evolvable Machines, vol 6 - 2, (2005) 3191-220.

[14] Kohmoto K., Katayama K., Narihisa H., "Performance of a genetic algorithm for the graph partitioning problem", Mathematical and Computer Modelling, 38; 11-13 (2003) 1325-1332.

[15] Lawler E., Lenstra J., Rinnooy Kan A., Shmoys D., "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", New York, Wiley, (1985).

[16] Lewis R.R.. "Simulated annealing for profile and fill reduction of sparse matrices", International Journal for Numerical Methods in Engineering, 37, (1994), 905-925.

[17] Lim I.L., Johnston I.W., Choi S.K, "Comparison of Algorithms for Profile Reduction of Sparse Matrices", Computers & Structures 57;2 (1995) 297-302.

[18] Mallya J.U., Zitney S.E., Choudhary S., Stadtherr M.A., "Matrix reordering effects on a parallel frontal solver for large scale process simulation", Computers and Chemical Engineering 23, (1999) 585-593.

[19] Mattfeld D.C., "Evolutionary search and the job shop. Investigations on genetic algorithms for production scheduling", Springer-Verlag (1995).

[20] Marti R., Laguna M., Glover F., Campos V., "Reducing the bandwidth of a sparse matrix with tabu search", European Jour. of Operational Research, 135; 2 (2001) 450-459.

[21] Radicati G., Vitaletti M. "Sparse Matrix-Vector Product and Storage Representations on the IBM-3090 with Vector Facility", Report 6513-4098, IBM-ECSEC, Roma, (1986).

[22] Van der Vorst H.A., "Bi-CGSTAB a fast and smoothly converging variant of Bi-CG for the solution of nonsym. linear systems", SIAM J. Sci. Statist. Comput, 13 (1992) 631-644.

[23] Winter G., Greiner D., Galván B., González B., 'Economical and Environmental Electric Power Dispatch Optimization', *Fifth Conference on Evolutionary Methods for Design, Optimization and Control with Application to Industrial and Societal Problems*. EUROGEN 2003. Ed: G. Bugeda, J.A. Desideri, J. Periaux, M. Schoenauer and G. Winter. CIMNE, Barcelona.