Unstructured 3-D Grid Partitioning Methods Based On Genetic Algorithms

K.C. Giannakoglou and **A.P. Giotis** [1]

Abstract. In this paper, two methods that are capable to efficiently partition 3-D unstructured grids for parallel processing, will be presented. Implicit to either of these methods are a recursive bisection pattern, a multilevel acceleration scheme and local refinement heuristics. Although both are based on Genetic Algorithms (*GAs*), their concept and formulation are completely different. The first method uses binary coding, where 0's and 1's denote owning processor for each grid entity. The second method encodes a few control parameters that define a unique bisection of the grid, on the equivalent graph, where a physical problem is solved. Advantages and disadvantages of both methods will be cited and a hybrid scheme will be suggested as a fast alternative partitioner.

1 INTRODUCTION

The numerical solution of large-scale physical problems is time-and memory-consuming. For an unstructured grid to be effectively processed on a distributed memory parallel computer, balanced grid subsets, as many as the available processors, should be created. "Good" Unstructured Grid Partitioning (*UGP*) methods should meet the following requirements

- to create subdomains with the same number of grid elements (tetrahedra), ensuring thus equally loaded processors
- to minimize the number of interface elements (triangular faces), ensuring thus an as low as possible interprocessor communication.

For evident reasons, the UGP software should not be CPU-time or memory- consuming. Note that, the aforementioned requirements correspond to non-overlapping grid partitions, but similar statements can be also made for overlapping partitions, provided that the evenly distributed grid entities are nodes rather than tetrahedra.

Among existing UGP methods, Recursive Spectral Bisection (RSB) [5] is by far the most popular. As the grid size increases, RSB becomes time-consuming and acceleration techniques [7] or local refinement heuristics [1] are employed. However, our intention is to refrain from comparing RSB and the proposed methods in terms of computer cost, since the cost of RSB partitioners depends highly upon the eigensolvers used.

The first proposed UGP method (to be referred to as *BITMAP*) is an extension of a 2-D grid partitioner that the authors presented in [4], while the second one (*FIELD*) is a novel method. The chosen abbreviations reflect the spirit of each method. Common features of the two partitioners are listed below:

1. They operate on the equivalent graph, which is formed by nodes (corresponding to tetrahedral elements) and links (corresponding to triangular faces). So, although 3-D grids with tetrahedral elements will be partitioned, the methods' extension to grids with different element types is straightforward.
2. They are based on the concept of recursive bisections, giving rise to 2^n subdomains.
3. The genetic operators are coupled with a single-pass multilevel scheme and upon convergence at any level, the *GAs* continue to operate at the next finer level.
4. At lower levels, special refinement algorithms improve the quality of the partition.

2 PROBLEM FORMULATION

Due to their recursive character, both methods will be formulated for a single bisection. By repetively applying the same algorithm, 2^n partitions can be obtained.

[1] National Technical University of Athens, Lab. of Thermal Turbomachines, P.O. Box 64069, 15710, Athens, Greece, kgianna@central.ntua.gr, agiotis@central.ntua.gr

ECCOMAS 98.
Published in 1998 by John Wiley & Sons, Ltd.

2.1 THE *BITMAP* FORMULATION

The *BITMAP* formulation is based on the mapping of the equivalent graph nodes to a bit-string, which is suitable for the selected optimizer (*GAs*). Each bit corresponds to a single graph node and its bit value determines whether this node belongs to the one or the other partition.

In *GAs*, the performance of each bit-string representing a candidate solution, is expressed by its fitness score. A fitness function incorporates the requirements of load balance and minimum interface size and returns a single value for each bit-string. In [4], a mathematical expression for the fitness function, that proved to be effective in 2-D UGP problems has been proposed. Its extension to 3-D grids is straightforward. Assume that N_1 and N_2 are the numbers of tetrahedra in the two subdomains, while N_{cf} is the number of faces along the interface. Then

$$C_1(N_1, N_2, N_{cf}) = \frac{|N_1 - N_2|}{\sqrt{N_1 N_2}} + w\frac{N_{cf}}{N_f} \qquad (1)$$

where N_f is the total number of graph links. The first quotient in eq. 1 is a measure of the load balance, whereas the second measures the interface size. The w parameter is a weighting factor, but for a discussion about w the reader should refer to [4].

2.2 THE *FIELD* FORMULATION

In the so-called *FIELD* formulation, the unstructured grid is mapped onto a 3-D graph, with graph nodes located at the barycentres of grid tetrahedra. This mapping gives rise to a parallelepiped domain which encloses the 3-D graph, with its six faces being parallel to the cartesian planes. A pair of point-charges is allowed to float within this domain. Each point-charge (A or B) creates a 3-D scalar field around it. The field value at any graph node (x, y, z) can be determined by considering its distances from $A(x_A, y_A, z_A)$ and $B(x_B, y_B, z_B)$

$$r_A = \sqrt{(x_A - x)^2 + (y_A - y)^2 + (z_A - z)^2}$$
$$r_B = \sqrt{(x_B - x)^2 + (y_B - y)^2 + (z_B - z)^2} \qquad (2)$$

in a way which mimics the static electric field around a charged particle. As in the Coulomb's law, two coefficients (k_A and k_B) need to be introduced. Two possible forms of the mathematical expressions for the scalar field value (potential) at any graph node (x, y, z) are given below:

$$F(x, y, z) = +e^{k_A r_A} - e^{k_B r_B} \qquad (3)$$

or

$$F(x, y, z) = +\frac{k_A}{r_A^2} - \frac{k_B}{r_B^2} \qquad (4)$$

The exponential decaying law in eq. 3 requires k_A and k_B to be negative. For the sake of convenience, negative k_A and k_B values can be used in eq. 4, as well. Of course, expressions 3 and 4 could be replaced by other "physical laws" with similar characteristics. A bisection obtained using this formulation is shown if fig. 1.

As follows from the previous discussion, eight parameters $(x_A, y_A, z_A, k_A, x_B, y_B, z_B, k_B)$ are sufficient to define a unique scalar field over the graph space. Sorting the graph nodes by the local scalar field values, orders the graph in a way that allows its bisection to be readily carried out; it is evident that, half of the graph nodes with the lower $F(x, y, z)$ values should belong to the first, while the remaining half to the second partition. In this way, the load balance requirement is automatically satisfied and the fitness function, needs to account only for the minimum communication requirement, thus

$$C_2(N_{cf}) = N_{cf} \qquad (5)$$

Note that the coefficient k for one of the charges may be defined arbitrarily. So, we will assume that $k_A = -1$ and the number of free-parameters is reduced by one !

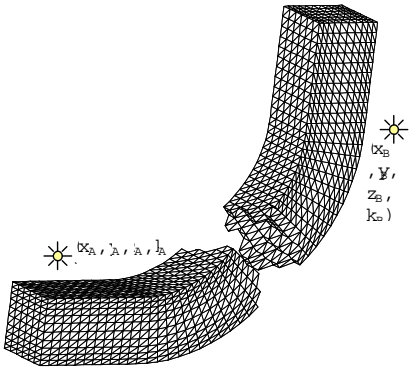

Figure 1. A single bisection, obtained using the *FIELD* formulation

3 OPTIMIZATION THROUGH *GAs*

GAs [2], [6] are based on the natural evolution and the Darwinian strife for life. Optimization is implicit to the NP-complete UGP problem and, as a consequence, *GAs* are candidate solution tools [3].

The *UGP* problem, as encoded in the *BITMAP* formulation, allows for a direct optimization through *GAs*.

On the other hand, the small number of free-parameters appearing in the *FIELD* formulation makes the use of *GAs* very attractive for this method as well. In the *FIELD* method, the seven free-parameters

$$(x_A, y_A, z_A, x_B, y_B, z_B, k_B) \qquad (6)$$

are first transformed to binary strings and are then concatenated to yield the full chromosome.

The basic *GAs* operators like crossover, parent selection and mutation, are quite the same in both methods, with the exception of some extra mutation operations of corrective character, applied in *BITMAP*. The crossover operator is a standard two-point one. A hybrid scheme for the parent selection is employed, where 90% of the individuals are selected through linear fitness ranking and the rest of them via probabilistic tournament. This allows a limited number of poor individuals to survive.

In [4], the authors modified some of the operators to accelerate the convergence of the *GAs*. The same modifications have been transferred to the 3-D *BITMAP* method. So, the mutation operator allows a limited number of bits to alter from 0 to 1 or vice-versa *(one-way scheme)*, at random with variable probability, according to the position of the corresponding gene in the current graph.

Over and above, in the *BITMAP* method, a corrective operator is applied to a small percentage of the population every n (for instance, $n = 7$) generations. Its purpose is to alter the owners of small groups of graph nodes that are completely enclosed by the rival owner. This correction step is associated with an increased probability (about 70%). Often, the so-corrected individuals become quite unbalanced, though with smaller interface size. The parent selection algorithm, ensures that some of these provisionally unbalanced individuals do survive in the new generation.

4 MULTILEVEL APPROACH AND LOCAL REFINEMENT

The multilevel approach is used in both methods. At each bisection , the initial graph G^0 (corresponding to the unstructured grid) becomes coarser by collapsing neighboring graph nodes into groups. The coarsening algorithm should avoid excessive deviations in groups' population. So, at level $G^m, m \neq 0$ a graph node will collect several G^{m-1} graph nodes. The graph node with the smaller population (in terms of the initial G^0 nodes it possesses) is considered first and nodes in contact with it are attached until its population number exceeds a pre-specified number proportional to the level m and to the initial grid size. A succession of M similar coarsening steps provides the final or coarser graph (highest level, G^M) with a size that can be easily handled by *GAs*. The G^M graph is first partitioned in two subsets yielding thus the initial chromosomes for the G^{M-1} graph bisection. This algorithm is repeated up to G^0.

The gain from the use of the multilevel approach is considerable. In the *BITMAP* method working with a coarser graph, makes chomosomes extremely shorter. The reduced length of chromosomes accelerates the genetic operators a lot, and combined with local refinement at the lower levels (see next section) yields a fast UGP software. In the *FIELD* method the chromosomes' length does not change from level to level, since the seven free-parameters are always encoded. However, economy in CPU-time is achieved since, at the higher levels all scalar field computations are performed on a coarser graph.

In the *FIELD* method the mutation operator described in the previous section has been modified to account for coarse graphs. So, at level $G^m, m \neq 0$ graph links (instead of graph nodes) are attributed a mutation probability proportional to the number of constituent faces and, in a second phase, one out of the two graph nodes sharing this link may alter according to the aforementioned *one-way* scheme. At level G^0, mutation probabilities are assigned to tetrahedra in proportion to the number of interface faces they possess.

At the lower levels, massive alternations in the partitions formed so far are not expected, so only a narrow zone of graph nodes in the vicinity of the interface needs to be handled by the optimizer. In this way, the computing cost decreases without damaging the quality of the partition.

5 THE *HYBRID* METHOD

In general, the *FIELD* method has less memory requirements than the *BITMAP* one, since a set of seven real variables, corresponding to the free-parameters in 6, suffices to define a grid partition. In contrast to the *BITMAP* method where the starting individuals generally correspond to messy partitions, the *FIELD* algorithm starts with partitions that are well in order, as defined by the randomly selected set of free-parameters. Nevertheless, the *FIELD* algorithm is associated with increased computing cost per generation due to the repetitive calculations of the scalar field according to eqs. 3 or 4 (3 is 25% slower than 4) and the sorting procedure. A worthnoting disadvantage of the same algorithm is that it becomes slow and often ineffective during the refinement of an almost optimum solution. At the refinement phase

the *BITMAP* method is very effective.

In view of the aforementioned advantages and disadvantages of the two developed methods, their coupling has been tested and proved to be an efficient and effective alternative for 3-D grid partitions. The so-call *HYBRID* method utilizes the *FIELD* algorithm during the first levels and switches to the *BITMAP* algorithm during the remaining ones. The corresponding software includes routines for the transformation of encoded chromosomes from the *FIELD* to the *BITMAP* method. The flow chart of the *HYBRID* method is shown in fig. 2.

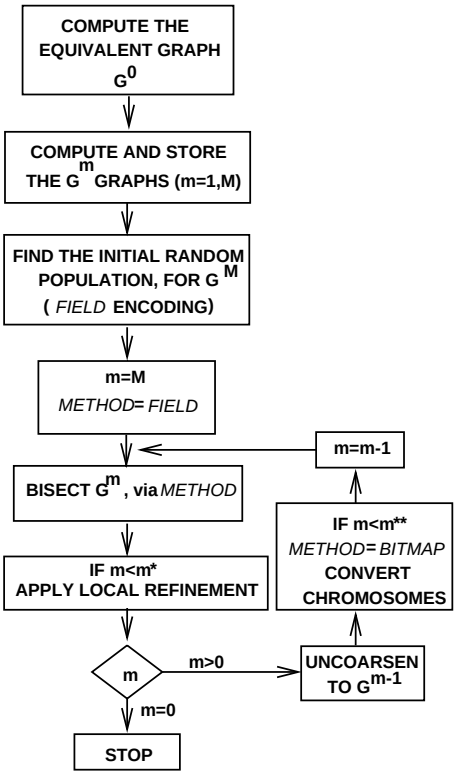

Figure 2. Flow chart of a single bisection for the *HYBRID* method

As it will become clear in the next section, the *HYBRID* method outperforms in terms of (a) memory requirements (vs. *BITMAP*), (b) computing cost (vs. both) and (c) quality of the resulting partitions (vs. both).

6 METHODS' APPLICATION

The methods presented were applied to the partition of three 3-D unstructured grids of different sizes (tabulated in 1), used in industrial CFD applications. These grids will be refered to as MESH1, MESH2 and MESH3. Each grid will be partitioned in up to $2^n = 32$ subdomains.

	MESH1	MESH2	MESH3
Tetrahedra	16560	37800	111440
Nodes	3619	9260	23122

Table 1. Sizes for the partitioned meshes

MESH3 for instance, stands for an unstructured grid, for flow calculations in a compressor cascade. A view of the obtained partitions of MESH3 into 32 subdomains is showed in Fig. 3. In this particular case, periodicity conditions at the graph level had to be applied. Since the proposed methods operate on the equivalent graph, it is straightforward to incorporate periodicity conditions, by introducing some extra graph links between tetrahedra in contact with the corresponding periodic faces.

Indicative *GA* parameters used to partition the three cases are listed in table 2.

Mutation probability is changing dynamically between the upper and lower bounds as the problem converges. Convergence is considered to occur when the average fitness score is not decreasing or a best-ever individual can not be found in N successive (idle) generations. In the *FIELD* method, seven bits per variable are used but this number should increase as the grid size increases in order to determine the location of point charges with more precision. Poor precision quickly explores the domain and results to a coarse interface in fewer generations. The generations done per bisection are in the range of $150 - 500$.

The parameters in *BITMAP* that greatly affect the results are the mutation probability and the weighting fac-

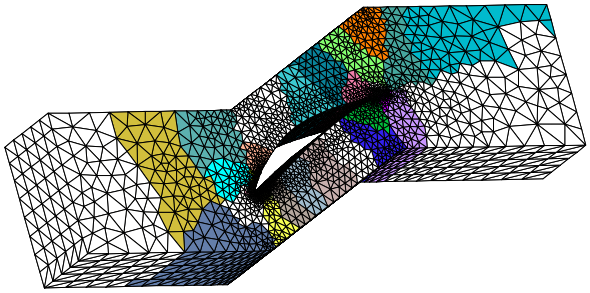

Figure 3. MESH3 partitioned into 32 subdomains with the *BITMAP* method.

Population size	50
Idle Generations	5
Mutation probability	0.35-0.7%
Crossover probability	90%
Max reproductive trials	3
Tournament size	2
Tournament probability	95%
Correction operates on	15% of population per 7 generations
Correction upper bound	40% of elements
Bits per variable	7

Table 2. *GAs* parameters

tor w. The latter varies, depending on the size of the graph to be partitioned. A recommended expression for w, which is the outcome of many numerical tests, is given below:

$$w \approx 8.96 \ln(N_1 + N_2) - 55 \qquad (7)$$

The grids' partition in 2^n subdomains, using the proposed methods, are tabulated in Tables 3. *GAs* parameters were kept constant aiming to a fair comparison, but smaller interface sizes can be obtained if these are fine-tuned. All methods resulted to balanced subdomains (within $\pm 0.3\%$ of the "perfect" balance) and so, we will not present any information concerning the load balance. The CPU-time results for each method and grid are incremental. The whole time for $2^4 = 16$ subdomains is the sum of the partial times up to 2^4. All times are in seconds and correspond to an Intel Pentium II 266MHz processor.

MESH1 INTERFACE					
	2^1	2^2	2^3	2^4	2^5
BITMAP	123	394	946	1572	2670
FIELD	132	394	946	1567	2600
HYBRID	127	383	904	1470	2504

MESH1 CPU-TIME (sec)					
	2^1	2^2	2^3	2^4	2^5
BITMAP	5	7	12	10	14
FIELD	20	16	15	17	12
HYBRID	6	5	9	8	11

MESH2 INTERFACE					
	2^1	2^2	2^3	2^4	2^5
BITMAP	252	764	1638	2554	3880
FIELD	310	830	1614	2614	4134
HYBRID	240	739	1489	2446	3853

MESH2 CPU-TIME (sec)					
	2^1	2^2	2^3	2^4	2^5
BITMAP	14	18	24	23	27
FIELD	54	84	61	53	44
HYBRID	12	11	25	19	23

MESH3 INTERFACE					
	2^1	2^2	2^3	2^4	2^5
BITMAP	204	891	1447	2321	3874
FIELD	330	1510	2698	4438	7320
HYBRID	198	853	1618	2488	3936

MESH3 CPU-TIME (sec)					
	2^1	2^2	2^3	2^4	2^5
BITMAP	28	37	27	37	53
FIELD	129	95	89	92	80
HYBRID	27	30	26	29	40

Table 3. Interface and CPU-time results for each method and grid for each bisection level 2^n

.

7 DISCUSSION

- Various alternatives of the *FIELD* method can be established. For instance, more than two-point charges can be used to drive a single bisection. This will expand the search space which will include more candidate solutions in conformity to a more general mathematical model, in the expense of increased chromosome lengths. The latter will increase the required number of generations and the computing cost per generation, since more scalar field computations should be performed. Moreover, the quality of the obtained partitions is in general similar to the one with two point-charges.
- Using an algorithm for the selection of the median in the *FIELD* method rather than using an indexing and ranking algorithm can further diminish the total CPU-time of *FIELD* (about 20%). However, the intermediate partitions at levels $G^m, m > 0$ are not exactly balanced since the number of tetrahedra collapsed in a graph node is not constant. However, the selected coarsening procedure is capable of creating groups with small deviations in the number of constituent tetrahedra. So, the aforementioned technique could (and has been) used instead; the results are almost identical to those obtained through the standard indexing algorithm, since the targeted balance is practically achieved at the finer level G^0.

8 CONCLUSIONS

The partitioning of unstructured grids for parallel processing is considered as an optimization problem, according the *BITMAP* and the *FIELD*formulations. Genetic Algorithms have been employed to solve the optimization problems and proved to be efficient and effective. Major remarks for each method are capitulated below:

1. The *BITMAP* method is a fast partitioning tool, generating refined interfaces, provided that the special genetic operators are employed. The encoding scheme increases the memory requirements, compared to the one used in the *FIELD* method. The two major parameters that affect the partition quality are the value of the weighting parameter w and the mutation probability.
2. Reduced memory requirements are associated with *FIELD* method. However, the evaluation of candidate solutions (calculation of the scalar field values at graph nodes and sorting) is time-consuming. Surprisingly, the careful selection of the mathematical law defining the scalar field at the graph nodes, affects the partition cost. As the grid size increases, the length of the bit-string associated with each one of the free-parameters should increase.
3. For both methods the local refinement during the lower levels and the multilevel treatment of the grids and the associated graphs is mandatory, otherwise the computing cost increases dramatically.

Considering the good refinement properties of the *BITMAP* method and the good initial partitions that the *FIELD* method may provide, a hybrid scheme was proposed. In this scheme, the *FIELD* method undertakes the partition during the first levels (coarser graphs) and the *BITMAP* method during the remaining ones, always in the context of the multilevel scheme.

ACKNOWLEDGEMENTS

The authors are indebted to Dassault Aviation for funding this research. They would also like to thank Prof. J. Periaux for stimulating the extension of the 2-D grid partitioner to 3-D grids.

REFERENCES

[1] Kernighan B.W. and Lin S., 'An efficient heuristic procedure for partitioning graphs', *The Bell System Techical Journal*, **49**, 291–308, (1970).

[2] Goldberg D.E., *Genetic Algorithms in search, optimization & machine learning*, Addison-Wesley, 1989.

[3] Von Laszewski G. Intelligent structural operators for the k-way graph partitioning problem. Proceedings of the Forth International Conference on Genetic Algorithms, 45-52, Morgan-Kaufman, 1995.

[4] A.P. Giotis and K.C. Giannakoglou, 'An unstructured grid partitioning method based on genetic algorithms', *Advances in Engineering Software*, (1998).

[5] Simon H.D., 'Partitioning of unstructured problems for parallel processing', *Computing Systems in Engineering*, **2(2/3)**, 135–148, (1991).

[6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, 2nd edn., 1994.

[7] Barnard S.T. and Simon H.D., 'A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems', *Concurrency: Practice and Experience*, **6**, 101–107, (1994).