EdgePack: A parallel vertex and node reordering package for optimizing edge-based computations in unstructured grids

Marcos Martins, Renato Elias and Alvaro Coutinho

Center for Parallel Computations and Department of Civil Engineering
Federal University of Rio de Janeiro, P. O. Box 68506,
RJ 21945-970 – Rio de Janeiro, Brazil
{marcos, renato, alvaro}@nacad.ufrj.br
http://www.nacad.ufrj.br

Abstract. A new and simple methodology is proposed to choose the best data layout for codes using iterative solvers in unstructured grid problems. This methodology is realized as a suite of routines named EdgePack, acting during pre-solution and solution phases, based on data locality optimization techniques and variations of the matrix-vector product algorithm. Results have been demonstrating the great flexibility and simplicity of this methodology, which is suitable for distributed memory platforms in which different data configurations can coexist.

1 Introduction

The performance optimization of codes based on iterative solvers for unstructured grids based problems has the matrix-vector product algorithm as a key issue. It is well known that data reordering techniques comprise an effective solution for good performance results during matrix-vector product computations, due to data locality optimizations [1]-[12]. Despite these techniques, overall performance can be further increased by the migration from element based to edge based data structure, as has been seen in the last decade [11], [13]-[18]. This data structure is quite suitable for reordering manipulations, strengthening even more the overall performance [19].

The combination of data reordering techniques with edge-based data structures generates lots of possibilities which can perhaps be the best performance solution for one or another hardware and software platform [19]. The code performance depends on many factors namely, computational architecture, compiler options, data configuration, number of degrees of freedom per node (algorithm complexity) and algorithm structure itself. According to previous results [19], there is no ultimate data combination known prior to the processing phase unless a probe is performed, since the many combinations possible may produce unexpected results.

This work presents a methodology to determine which data combination for a given computer platform is the best one in terms of processing time. This methodology is simply based on the choice of the best results after probing the several possibilities more adequate to such platform. Similar ideas have been implemented for dense matrix computations [20]. Considering a particular parallel platform, as clusters (heterogeneous or not), the best data configuration choice can be different from each other and thus, different data structures can be used together for the same model. Concerning distributed parallel processing, the standard approach is to partition data before performing data reordering on each processor. Data partition is performed by the Metis library [21].

Based on the nodal renumbering algorithms and concepts proposed in [4], [22], [23] and edge renumbering algorithms proposed in [17], algorithms for matrix-vector product composed by 1, 3 and 4 degrees of freedom per node (hereafter, referred to as *dof*), for symmetric and non-symmetric matrices, were implemented. The techniques employed try to determine the most suitable data reordering according to the computational system at hand, even without any prior knowledge about the processors architecture. Among them, a sorting of edges, in increasing order by the edge first node number (namely hereafter, reduced indirect addressing or reduced i/a), halves the indirect addressing operations of the edge-based matrix-vector product algorithm [22].

Additionally, data locality algorithms were used in association with special edge groupings, which improve the edge-based matrix-vector product algorithm. This was implemented for tetrahedra, grouped into 3 and 6 edges, named respectively superedge3 and superedge6 for both incompressible fluid flow and geomechanics problems [17]. For the latter, an edge-based interface element was implemented, with special groupings named superedge4 and superedge9, comprising groups of 4 and 9 edges respectively [24].

Related to memory dependency, lists of edges are built where no pair of nodes in the same edge list shares the same node. This arrangement is referred here as nodal disjointing and is responsible for a significant lack of performance. A Reverse Cuthill McKee (RCM) [25] algorithm in conjunction with edge and element sorting according to node numbering are further employed to diminish the negative effect provided by the nodal disjointing ordering.

The methodology developed is realized as a suite of routines built in Fortran90, comprising the necessary tasks for both pre-solution and solution phases, for serial or parallel platforms (shared, distributed or hybrid), named EdgePack. The remainder of this work is organized as follows. In the next section, we revise the edge-based data structures and the data reordering algorithms. In Section 3 we describe EdgePack in detail. In Section 0, we show some numerical experiments exploring the main characteristics of EdgePack. The paper ends with a summary of our conclusions and remarks.

2 Edge-Based Structure and Data Reordering

Edge based finite element data structures have been introduced for explicit computations of compressible flow in unstructured grid finite element and finite volume computations [13], [14], [26]. It was observed in these works that residual computations with edge-based data structures were faster and required less memory than standard element-based residual evaluations. Following these ideas, Coutinho *et al* [15], Catabriga and Coutinho [16], Sydenstricker *et al* [24], Elias *et al* [18] derived edge-based finite element implementations respectively for elasto-plasticity, the SUPG finite element formulation with shock-capturing for inviscid compressible flows, interface elements for modeling joints and faults and the SUPG/PSPG solution of incompressible flows. Differently from the previous finite volume/finite element implementations, all these works used the concept of algebraically disassembling the finite element matrices to build the edge matrices introduced by Catabriga and Coutinho [16]. This procedure makes the edge-operators construction independent of the underlying finite element formulation. To illustrate this procedure, consider three dimensional problems on unstructured meshes composed by tetrahedra. Thus, the element matrices can be disassembled into their edge contributions as

$$\mathbf{A}^e = \sum_{s=1}^{m} \mathbf{T}_s^e \ . \tag{1}$$

where $\mathbf{T}_s^e$ is the contribution of edge s to $\mathbf{A}^e$ and m is the number of edges per element. The contributions of all elements sharing a given edge s is given by the following matrix,

$$\mathbf{A}_s = \sum_{s \in \mathsf{E}} \mathbf{T}_s^e \ . \tag{2}$$

where E is the set of all elements sharing a given edge s.

When working with iterative solvers, it is necessary to compute sparse matrix-vector products. These matrix-vector products are responsible for a good share of the overall computational effort. A straightforward way to implement the edge-by-edge, similar to popular element-by-element matrix-vector product is

$$\mathbf{Ap} = \sum_{l=1}^{ne} \mathbf{A}^l \mathbf{p}^l \ . \tag{3}$$

where ne is the total number of local structures (edges or elements) in the mesh and $\mathbf{p}^l$ is the restriction of $\mathbf{p}$ to the edge or element degrees-of-freedom. As a prototype of such procedure, the edge-based Laplacian loop algorithm (comprising 1 dof per node) as proposed in [22], is:

Laplacian loop for a single edge sparse matrix-vector product

```
do edge = edge_begin, edge_end
   eq_1 = lm(1,edge)
   eq_2 = lm(2,edge)
   ap = a(edge) * (u(eq_2) - u(eq_1))
   p(eq_1) = p(eq_1) + ap
   p(eq_2) = p(eq_2) + ap
end do
```

where array `lm` stores edge equation numbers, `a` stores the edge coefficient and `u` stores the unknown values.

In order to achieve a good balance between memory accesses and floating point operations (*flops*), reordering techniques are suggested in the literature such as *superedges* [17], [27] and reduced indirect addressing edges [22]. The *superedge* scheme reaches good balance of i/a and *flops* [15] without complex preprocessing codes [17]. In this case, computer costs are just related to the new order of the edge list, considering the edges agglomerated, in geometric a sense, for example, in tetrahedral shape, swept by stride of 6 edges.

An alternative to reduce i/a is to convert an edge-based loop into a vertex-based loop [4] in which the edges are arranged in such a way that the first node always has the lower number and the first node number increases as the edge number increases with stride one. This loop reuses vertex-based data items in most or all of the accesses several times before discarding it. This approach increases flops but reduces i/a operations, whereas the edge has to be processed twice. **Table 1** shows a comparison of computation parameters for the matrix-vector multiplication algorithm for reduced i/a and superedge schemes, considering 3 *dof* and symmetric operators.

Table 1. Comparison of computational parameters for matrix-vector multiplication algorithm for reduced i/a and *superedges* for 3 *dofs*

Group/Parameter	flops	i/a	Flops/(i/a)
Simple Edge	36	18	2.0
*Superedge*6	268	36	7.4
*Superedge*3	130	27	4.8
*Superedge*9	436	54	8.1
*Superedge*4	190	36	5.3
Reduced i/a	39	9	4.3

The choice among all these different possibilities and data layouts, trying to minimize indirect addressing and improve data locality is not easy. In the following Section we introduce EdgePack, which contains heuristics to optimize these parameters.

3 EdgePack

EdgePack is a suite of routines built in Fortran90 to optimize computations on unstructured grids. EdgePack is divided into two sets of routines: the first set is a preprocessing phase and aims to reorder finite element meshes composed of tetrahedra and prisms to improve the performance of iterative solvers for any number of degrees of freedom per node. The main concern is to optimize data locality and data reuse for serial or parallel shared or distributed memory computers. The second set is composed of a series of optimized edge-based data routines for matrix-vector products and element matrix disassembling into edges.

The main tasks performed by EdgePack during the preprocessing phase comprise the edge connectivity assembly, based on building fast hash-tables, edge groupings into *superedges* [17], [27], nodal and edge reordering into reduced indirect addressing edge mode [4], [22], nodal reordering to minimize bandwidth based on the Reverse Cuthill McKee (RCM) algorithm [25], nodal disjointing reordering for pipelined processing providing data with no memory dependencies, edge reordering driven by equation map and element reordering according to edge connectivity.

EdgePack runs either on serial or parallel distributed memory systems. Targeting on parallel processing, the mesh partitioning is performed by the Metis library [21], in weighted or non-weighted mode, and for all subdomains data reordering is accomplished locally on each processor node. This methodology provides the possibility of achieving the best data structure and reordering choice for each processing node, as in the case of heterogeneous clusters. EdgePack probes timing results for matrix-vector products based on equation map and element matrix topology, taking into consideration the number of degrees of freedom per node and if the matrix is symmetric or not, and chooses the data configuration from the best matrix-vector product timing result. Based on this probe, it is possible to determine which data structure will suit best on a given hardware and software configuration and automatically decide which element, edge and node data structure and order fit best on, without user concern or intervention. However, the user can set directly which data structure to use without probing. EdgePack can be used as either a stand-alone program or library.

Communication among processors is another issue treated by EdgePack. The subdomain interfaces, inherent in distributed parallel processing, can either be done by simply indexing shared nodes among subdomains – thus preserving local data order – or by ordering shared nodes sequentially for optimizing communications tasks.

For the processing phase, EdgePack provides optimized edge-by-edge matrix-vector product routines for 1 up to 4 *dof* per node, for symmetric or non-symmetric operators, which account for i/a and *flops* reduction, and data reuse strategy based on data locality and agglomeration into registers. Besides data configuration paradigm, EdgePack probes the matrix-vector product routines based on typical vector lengths for chunkwise and nodal disjointed loops, alternative right hand side evaluation (RHS) [22] and loop unrolling into edges. The matrix-vector product routines are ready to run under serial and parallel (hybrid or not) mode and are set according to pre-compiler directives.

The various data configurations and matrix-vector product modes available can be set by a combination of 7 keywords as presented by **Table 2**. The first five keywords are related to the preprocessing phase and the remaining ones set the matrix-vector product algorithm type.

Table 2. Keywords for data and matrix-vector algorithm configuration

Keyword #	Description	Option
1	Nodal Disjointing	Yes / No
2	Chunks	Yes (List Length) / No
2.1	List Length	Starting from 64 up to 2048 and free (mandatory for reduced 1 and 2)
3	Node Order	RCM / Reduced
4	Edge Order	Reduced (0/1/2) / Simple or *Superedge*
4.1	Prescribed Edge	Yes / No (for simple and *superedge* only)
4.2	*Superedge* Omission	-Sx (for *superedge* only, x = 3, 4, 6 and 9)
5	Shared Nodes (for distributed processing)	Indexed / Sequential
6	Alternative RHS	Yes / No
7	Loop Unrolling	Yes (2/3/6) / No

The following section presents examples of EdgePack features.

4 Preliminary Results

This section presents some preliminary results from three models comprising geomechanics and incompressible fluid flow. The first two are models of true sedimentary basins with faults. The last model comprises the transient incompressible fluid flow around a cylinder. The first model illustrates the main data orderings available in EdgePack, for serial processing, through edge connectivity graphs, characterizing data locality for each case. In the second model we show time probing results for some data ordering available on EdgePack for serial mode, searching the fastest ones for each case among the various data configuration possibilities and related results. The third model presents edge connectivity graphs, timing results for the best data configuration selection after probing, besides some validation results for parallel processing.

4.1 Sedimentary Basin – Model 1

This model represents a sedimentary basin the geometry and material of which correspond to a region in the Colombia, South America. A fault is present in the model, crossing it completely. The model surface is approximately 80 × 80 km^2 and 23 km deep, and comprising 141,766 tetrahedra, 5,133 interface elements and 28,897 nodes. **Fig. 1** presents the surface of the sedimentary basin mesh – model 1.

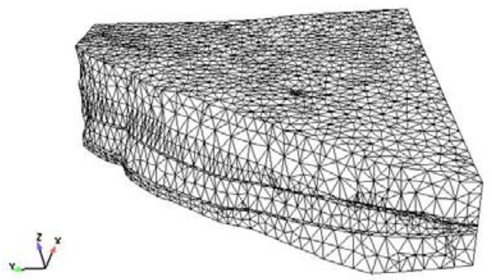

Fig. 1. Surface mesh of sedimentary basin - model 1

The model will be employed to illustrate the main data orderings available in EdgePack. **Fig. 2** presents the main edge and vertex ordering generated by EdgePack for this model by a representation of its edge connectivity, where the first edge node is in black line and the second one in gray line. The edge sequence is highlighted through lines connecting the nodes. In this figure, the original vertex and edge orderings are presented in panel (a); the reduced edge order is presented in panel (b) in which the monotonic order of the first nodes can be clearly noted by the ramps. Panel (c) presents the *simpleedge* order and panel (d), the *superedge* one. It can be noted on these orderings the good data locality configuration. In panel (d), the five ramps represent the *superedges* employed as *superedge*6 (S6), *superedge*3 (S3), *superedge*9 (S9), *superedge*4 (S4) and *simpleedge* respectively.

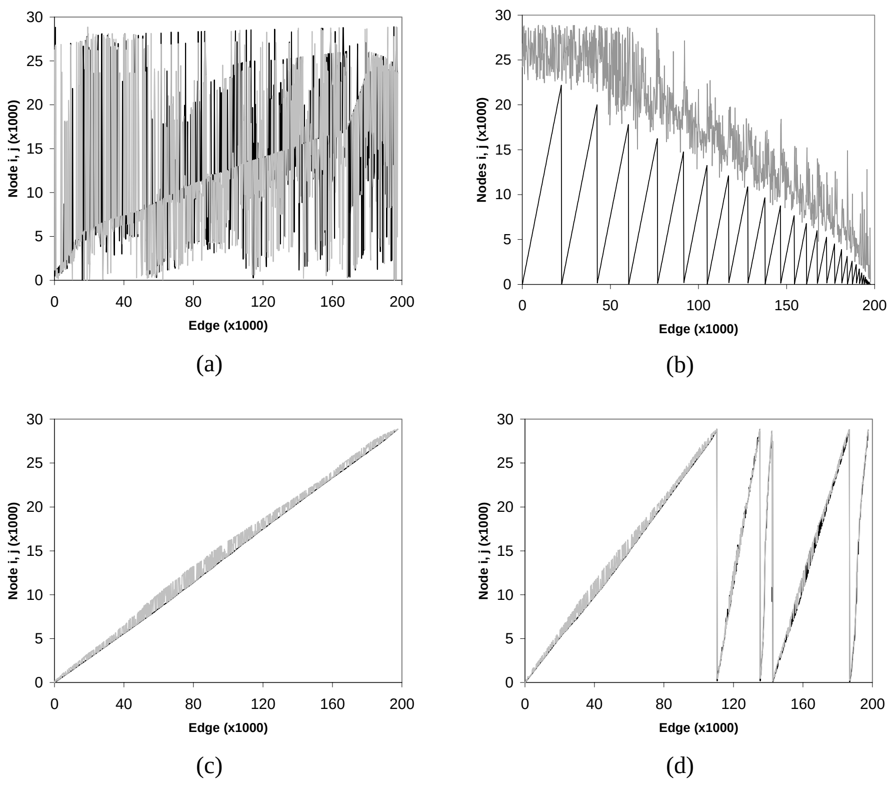

Fig. 2. Edge connectivity orders generated by EdgePack for sedimentary basin – model 1

4.2 Sedimentary Basin – Model 2

This model represents a portion of a sedimentary basin the geometry and material of which correspond to a region in the northeast of Brazil, South America. The model is constituted of four blocks separated by three geological faults. The model is 9.1 × 6 km^2 and 1.5 km deep, comprising 371,244 nodes, 2,064,940 linear tetrahedral elements and 17,317 interface elements, as shown in Fig. 5. The boundary conditions impose compression along the major dimension, normal to faults and normal displacements nullified over the entire surface, besides overburden from upper layers and self-weight. Analysis comprises 3 *dofs* per node as displacements.

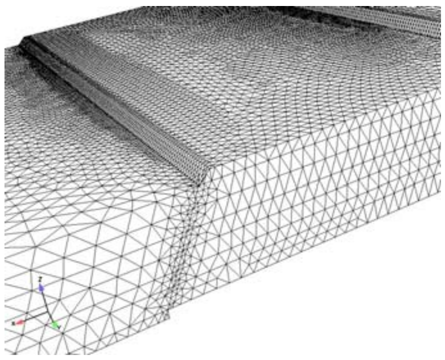

Fig. 3. Surface mesh of sedimentary basin - model 2

Fig. 4 presents the most important time probing results out of 208 jobs on an Itanium 2 platform, under pipelined serial mode. **Fig. 4**(a) presents some time probing results where the clear benefit over reduced schemes for this case is represented by a 45% gain in time. The labels correspondence is presented by **Table 3** in which results were obtained for non-unrolled matrix-vector loops.

Table 3. Data configuration for results presented by **Fig. 4**(a)

Label	Nodal Order	Chunk Length (min/max)
Simpleedge	RCM	64/2048
Superedge	RCM	64/ 512
Reduced 0	Reduced	1/ 64
Reduced 1	Reduced	64/ 64
Reduced 2	Reduced	64/ 128

Fig. 4(b) presents the percentage of occurrences of nodal order configuration for all 208 jobs. The percentage is referred for each legend individually. It is clear the advantage of RCM nodal ordering over reduced one for this combination of model and platform. However, only 4% of all cases with nodes ordered by RCM attain the best results.

For the reduced edge scheme, **Fig. 4**(c) presents the percentage of occurrences related to each one individually. This picture shows the slight advantage of reduced 0 and 1 scheme over reduced 2, for this case. However, the distribution tends to be uniform for three modes.

Fig. 4(d) pictures time probing results for nodes ordered by RCM and edges arranged as *simpleedges* and *superedges*. In the latter, all *superedges* available were used. The supremacy of *simpleedge* over *superedge* is clearly noticeable since *superedge* only occurs in the third time scale. However, only 7% of *simpleedge* data combinations appear as best results.

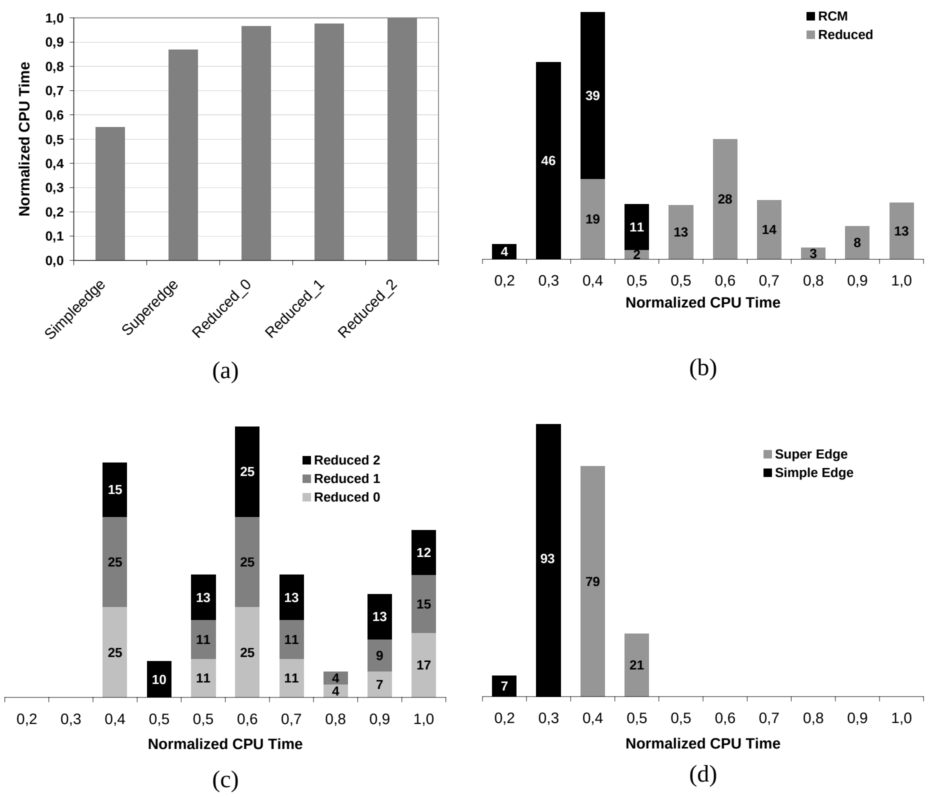

Fig. 4. Timing results from probing of sedimentary basin – model 2

4.3 Incompressible Fluid Flow

The problem of a fluid flowing around a circular cylinder is considered as an application of the EdgePack's data improvements to incompressible fluid flow codes. Time probing for this case was performed and the data configuration comprising edge ordering by the *simpleedge* scheme, nodal ordering by RCM and chunk length of 4096 was chosen. For this problem an extension for transient flows of the edge-based stabilized finite element implementation described in [18], is applied to solve the three dimensional u-p fully coupled (4 *dofs* per node) problem arising from the Navier-Stokes discretization. The computational domain following the dimensions described in [28] and the mesh formed by 446,662 linear tetrahedra elements, 1,010,367 edges, and 81,991, comprising 174,008 equations.

The results, for a Reynolds 100, are assessed and compared with those presented by [29] and [30] showing a good agreement for the time evolution of the drag and lift forces on the cylinder surface as depicted in **Fig. 5** for lift coefficient. Baranyi [30] reported a Strouhal number of S_t = 0.163 and drag and lift coefficients of 1.346 and 0.228 respectively for Reynolds 100 while Williamson [29] employing an experimental correlation for the Strouhal-Reynolds pair, estimated the value of 0.1643. In this work we have found S_t = 0.16 and 1.313 and 0.225 for drag and lift coefficients respectively which compares well with the results presented by those authors.

A typical computation of this problem considering 10,000 fixed time steps, which corresponds to 500 time units, spent 6.48 hours running in MPI mode with four processors of a SGI Altix 350 system equipped with Intel Itanium-2 1.5 GHz processors.

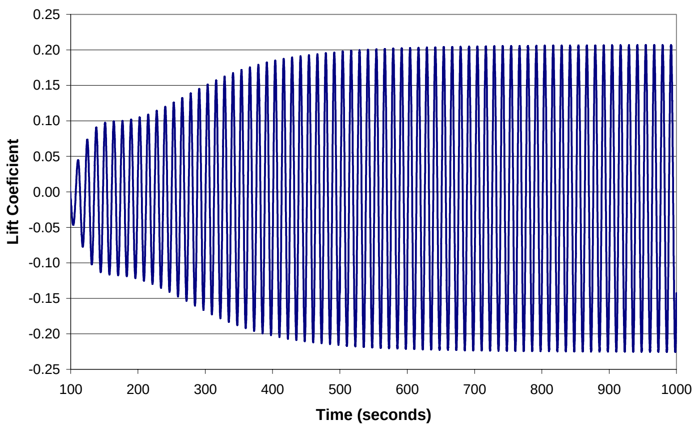

Fig. 5. Lift coefficient for flow around a cylinder problem

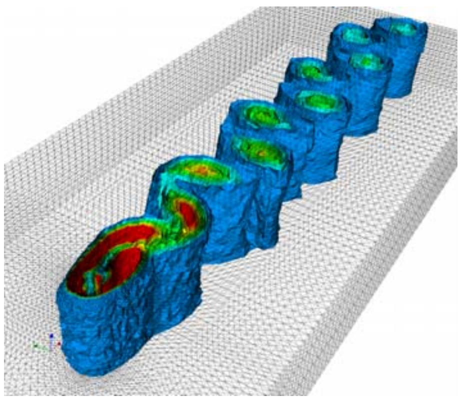

Fig. 6. Snapshot of cylinder surface mesh and vorticity, showing the development of the von Karmann vortex streets

Fig. 7 presents nodal edge connectivity for partition 2 out of 4 partitions. For this figure, panel (a) presents the original node order for edges ordered in chunks for pipeline processing and panel (b) corresponds to the combination of *simpleedge* order in chunks for pipelined processing with nodes renumbered by RCM. Computations with the node and edge order in panel (b) are twice as fast as those in panel (a).

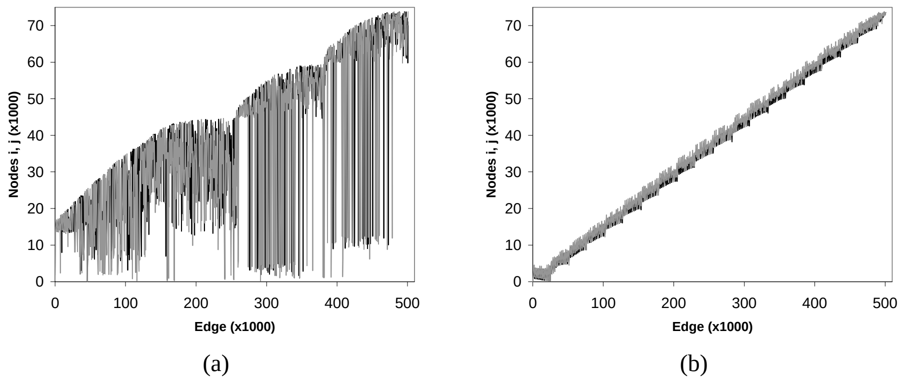

Fig. 7. Nodal edge connectivity of *simpleedge* for partition 2

5 Conclusions and Incoming Work

The results presented attest to the versatility of EdgePack in choosing data configuration according to best time results. This versatility can be derived from the need of determining which data configuration produces the best time performance among hundreds of possibilities besides computational platform effects.

The model presented in Section 4.1 demonstrated the main data configuration possibilities and its choice range. The mesh partitioning was also exploited for distributed parallel processing. The good effect over data locality was clearly shown by nodal edge connectivity graphs.

An example of time probing for a serial run was done in 4.2, where performance results were undetermined a priori and probing demonstrated that a wrong choice could represent about 45% loss in time processing. The presented graphs glimpsed the various possibilities and their unexpected results strengthening the EdgePack flexibility in setting data configuration without user intervention.

Section 4.3 presented an example of distributed parallel processing over partitions locally ordered by EdgePack. In this example, data was prepared to run in hybrid mode, comprising data distribution, memory dependency and good data locality.

As next step in EdgePack development, the work goes towards distributed parallel processing in heterogeneous clusters and grids, where in each node EdgePack can determine different data configuration, to set the best performance individually providing the coexistence of different data structures during same analysis and exploiting the most adequate data configuration for each processor.

References

1. Burgess, D. A. and Giles, M. B.: Renumbering unstructured grids to improve the performance of codes on hierarchical memory machines, Advances in Engineering Software 28 (1997) 189-201
2. Carey, G. F., Swift, S. and McLey, R. T.: Maximizing sparse matrix-vector product performance on RISC based MIMD computers. Journal of Parallel and Distributed Computing, v.37, p.146-158, 1996
3. Douglas, C. C, Hu, J., Kowarschik, M., Rude, U. and Weiss, C.: Cache Optimization for Structured and Unstructured Grid Multigrid. Electronic Transactions on Numerical Analysis, v.10, p.21-40, 2000
4. Gropp, W. D., Kaushik, D. K., Keyes, D. E. and Smith, B. F.: Performance Modeling and Tuning of an Unstructured Mesh CFD Application, Proceedings of SC 2000, IEEE Computer Society, 2000, Dallas, Texas, United States, Article No. 34, ISBN:0-7803-9802-5
5. Löhner, R.: Renumbering Strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines, Computer Methods in Applied Mechanics and Engineering 163 (1998) 95-109
6. Oliker, L., Canning, A., Carter, J., Shalf, J. and Skinner, D.: Evaluation of cache-based superscalar and cacheless vector architectures for scientific computations, Proceedings of the 18th Annual International Conference on Supercomputing, Malo, France, 2004, ISBN:1-58113-839-3
7. Oliker, L., Li, X., Heber G. and Biswas, R.: Parallel Conjugate Gradient: Effects of Ordering Strategies, Programming Paradigms, and Architectural Platforms, IEEE Transactions on Parallel and Distributed Systems, 11(9):931-940, 2000
8. Oliker, L., Li, X., Heber, G. and Biswas, R.: Ordering Unstructured Meshes for Sparse Matrix Computations on Leading Parallel Systems, Lecture Notes In Computer Science, Vol. 1800, pp. 497-503, 2000
9. Oliker, L., Li, X., Husbands, P. and Biswas, R.: Effects of Ordering Strategies and Programming Paradigms on Sparse Matrix Computations, SIAM Review, Vol. 44, No. 3, pp 373-393, 2002
10. Pinar, A. and Heath, M. T.: Improving Performance of Sparse Matrix-Vector Multiplication, Conference on High Performance Networking and Computing, Proceedings of the 1999 ACM/IEEE Conference on Supercomputing (CDROM), Portland, Oregon, United States, Article No. 30, 1999, ISBN:1-58113-091-0
11. Ribeiro, F. L. B and Coutinho, A. L. G. A.: Comparison between element, edge and compressed storage schemes for iterative solutions in finite element analyses. International Journal for Numerical Methods in Engineering, Volume 63(4): 569-588, 2005
12. Vuduc, R., Demmel, J. W., Yelick, K. A., Kamil, S., Nishtala, R., and Lee, B.: Performance Optimizations and Bounds for Sparse Matrix-Vector Multiply. Conference on High Performance Networking and Computing, Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, Baltimore, Maryland, Pages: 1 – 35, 2002
13. Peraire J, Peiro J, Morgan K, 1993. Multigrid solution of the 3d-compressible Euler equations on unstructured grids. Int. J. Num. Meth. Engrg.. 36(6): 1029-1044
14. Luo H, Baum JD, Löhner R, 1994. Edge-based finite element scheme for the Euler equations, AIAA Journal, 32(6):1183-1190
15. Coutinho ALGA, Martins MAD, Alves JLD, Landau L and Moraes A, 2001. Edge-based finite element techniques for non-linear solid mechanics problems. Int. J. for Num. Meth. in Engrg, 50(9):2053-2068
16. Catabriga L and Coutinho ALGA. , 2002. Implicit SUPG solution of Euler equations using edge-based data structures. Computer Methods in Applied Mechanics and Engineering, 32:3477-3490

17. Martins, MAD., Alves, JL., Coutinho, ALGA.: Parallel Edged-Based Finite Techniques for Nonlinear Solid Mechanics. Lecture Notes on Computer Science, Vol. 1981, Springer-Verlag, Berlin Heidelberg (2001), pp 506-518
18. Elias R N, Martins MAD, Coutinho ALGA, Parallel Edge-Based Inexact Newton Solution of Steady Incompressible 3D Navier-Stokes Equations, J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1237–1245, 2005
19. Coutinho ALGA, Martins MAD, Sydenstricker R and Elias RN. Performance comparison of data reordering algorithms for sparse matrix-vector multiplication in edge-based unstructured grid computations, International Journal for Numerical Methods in Engineering, Chichester, UK, v. 66, n. 3, p. 431–460, 2006
20. Whaley, RC, Petitet, A, Dongarra, J, Automated Empirical Optimizations of Software and the ATLAS Project, Parallel Computing 27(1–2):3–25, 2001
21. Karypis G. and Kumar V., Metis 4.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, (1998). http://www.users.cs.umn.edu/~karypis/metis
22. Löhner, R., Galle, M.: Minimization of indirect addressing for edge-based field solvers, Communications in Numerical Methods in Engineering, 18 (2002) 335-343
23. Löhner, R.: Some useful renumbering strategies for unstructured grids, International Journal for Numerical Methods in Engineering, Vol. 36, (1993) 3259-3270
24. Sydenstricker, R.M., Martins, M.A.D., Coutinho, A. L. G. A., Alves, J.L.D.: Edge-Based Interface Elements for Solution of Three-Dimensional Geomechanical Problems. Lecture Notes in Computer Science, v.2565, p.53 - 64, 2003
25. Cuthill, E. and McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In Proc. ACM Nat. Conf., pp 157-172, 1969
26. Barth, T. J., "Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes", AIAA, 29th Aerospace Sciences Meeting, January 7-10, AIAA 91-0721, Reno, Nevada, 1991
27. Löhner, R.: Edges, Stars, Superedges and Chains; Comp. Meth. Appl. Mech. Eng. 111, 255-263 (1994)
28. Kalro V. and Tezduyar T.E., Parallel 3D Computation of Unsteady Flows around Circular Cylinders, Parallel Computing 23 (1997) 1235-1248
29. Williamson, CHK, Defining a Universal and Continuous Strouhal-Reynolds Number Relationship for the Laminar Vortex Shedding of a Circular Cylinder, Phys Fluids 31 (1988) 2742-2744
30. Baranyi, L, Computation of Unsteady Momentum and Heat Transfer from a Fixed Circular Cylinder in Laminar Flow, Journal of Computational and Applied Mechanics, vol 4, no. 1, (2003) pp. 13-25