A Patch-Based Mesh Optimization Algorithm for Partitioned Meshes [⋆]

Nicholas Voshell[1], Suzanne Shontz[1], Lori Diachin[2], Patrick Knupp[3], and Todd Munson[4]

[1] Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
njv116@cse.psu.edu, shontz@cse.psu.edu
[2] Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551
diachin2@llnl.gov
[3] Applied Mathematics and Applications
Sandia National Laboratories
Albuquerque, NM 87185
pknupp@sandia.gov
[4] Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
tmunson@mcs.anl.gov

Abstract. The finite element method is commonly used in the numerical solution of partial differential equations (PDEs). Poor quality elements effect the convergence and stability of the method as well as the accuracy of the computed PDE solution. Thus, mesh quality improvement methods are often used as a post-processing step in automatic mesh

[⋆] The work of the first author was supported by a Lockheed Martin Graduate Fellow Award, a Harry G. Miller Graduate Fellowship, and The Pennsylvania State University. The work of the second author was funded in part by NSF grant CNS 0720749, a Grace Woodward grant from The Pennsylvania State University, and the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. Part of the work of the second author was performed while a member of the Center for Applied Mathematics at Cornell University and an intern at Argonne National Laboratory. The work of the third and fifth authors was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. The work of the third author was performed at Argonne National Laboratory. The work of the fourth author was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC-94AL85000.

generation.

Optimization-based methods have proven effective for mesh quality improvement and fall into two categories: local methods (which adjust one interior vertex position at a time) and global methods (which simultaneously adjust all interior vertex positions). We consider the intermediate case and propose a patch-based mesh optimization method (which decomposes the mesh into subset 'patches' to be optimized one at a time). We investigate the effects of the number of patches and the amount of patch overlap on mesh optimization time and mesh quality and compare our results with those obtained from the global and local versions of the method. Numerical experiments show that an implementation based on the feasible Newton algorithm is up to 69.3% and 92.9% faster than global and local vertex repositioning, respectively.

Key words: parallel scientific computing, mesh quality improvement, optimization, mesh partition

1 Introduction

Discretization methods, such as the finite element method, are commonly used in the numerical solution of partial differential equations (PDEs). The accuracy of the computed solution depends on the degree of the approximation scheme and number of elements in the mesh [1], as well as the quality of the mesh [2, 3]. In addition, the stability and convergence of the finite element method is affected by poor quality elements. Various techniques such as adaptivity [4], vertex repositioning [5, 6], and swapping [7] can be used to improve the quality of meshes resulting from automatic mesh generation, dynamic mesh motion, or physical simulation. This work focuses on vertex repositioning techniques for mesh quality improvement.

Optimization-based formulations of the mesh vertex repositioning problem use the mesh quality as an objective function [8–10]. Existing optimization-based methods for vertex repositioning are either local or global in nature. Local methods seek to improve the quality of the mesh by adjusting the position of one interior vertex at a time based on a local objective function, with convergence based upon a global objective function. They are most common in engineering practice. Global methods simultaneously adjust all interior vertex positions each iteration. They are gaining in popularity because of excellent performance obtained for some algorithms [11].

Freitag Diachin et al. [12] demonstrate numerous tradeoffs between local and global mesh vertex repositioning methods including: computational requirements, amount of memory required, time to solution, accuracy of the solution, and rate of convergence. Additional factors which may influence performance metrics include: the number of elements, element heterogeneity, element anisotropy, the type of mesh (structured vs. unstructured), the quality metric, the objective function, and the desired level of accuracy in the optimized mesh [12]. In [12], it was demonstrated that there is not a clear winner between the global and local versions of the feasible Newton (FeasNewt) algorithm [11].

Similarly, optimization-based methods that operate on an intermediate subset of the mesh, i.e., *a patch*, may produce optimal meshes in less time than the corresponding local or global versions of the algorithm by balancing the tradeoffs between the amount of memory required to calculate a step and the amount of additional knowledge available from neighboring vertices.

2 Problem Statement

2.1 Mesh Quality Metric

For this study, we employ the inverse mean-ratio mesh quality metric [13] which measures the difference between a trial element and an ideal element. Our description of the inverse mean-ratio metric is for a tetrahedral element and follows that of [11] and [14]; this derivation can be extended to other element types.

Suppose e is a tetrahedral element of an unstructured mesh M such that $x \in \mathbb{R}^{3\times 4}$ is a matrix containing the coordinates of each of e's four vertices x_1, x_2, x_3, and x_4. Define an incidence function $A : \mathbb{R}^{3\times 4} \to \mathbb{R}^{3\times 3}$ as follows:

$$A(x) := \left[x_2 - x_1 \quad x_3 - x_1 \quad x_4 - x_1\right],$$

where x_i denotes the i^{th} column of the matrix argument. Note that the volume of e is given by $\frac{1}{6}|\det(A(x))|$. Let $w \in \mathbb{R}^{3\times 4}$ denote the vertex coordinates for an ideal element (a unit equilateral tetrahedra for this study). Observe that $A(x)A(w)^{-1}$ is the identity matrix precisely when the trial element x and ideal element w have the same shape and size.

The inverse mean-ratio metric between x and w is given by

$$\frac{\|A(x)A(w)^{-1}\|_F^2}{3\det(A(x)A(w)^{-1})^{\frac{2}{3}}}.$$

The inverse mean-ratio metric is independent of scaling and takes on values between one (optimal) and infinity (degenerate) for positively oriented elements.

2.2 Mesh Quality Improvement Problem

Let V and E denote the set of vertices and elements in M, respectively, and V_I and V_B denote the set of vertices in the interior and (fixed) boundary of the mesh, respectively. Then, the problem of minimizing the average inverse-mean ratio over the entire mesh M becomes:

$$\min_{x\in\mathbb{R}^{3\times|V|}} \qquad \Theta(x) := \frac{1}{|E|}\sum_{e\in E}\frac{\|A(x_e)A(w)^{-1}\|_F^2}{3\det(A(x_e)A(w)^{-1})^{\frac{2}{3}}} \tag{1a}$$

$$\text{subject to} \qquad \det\left(A(x_e)A(w)^{-1}\right) > 0,\ \forall e \in E \tag{1b}$$

$$x_i = \bar{x}_i,\ \forall i \in V_B, \tag{1c}$$

where $\bar{x}_i$ denotes the fixed coordinates of the i^{th} boundary vertex. Constraint (1b) ensures that each element has a consistent orientation which is necessary for most discretization methods to work properly [15]. We solve the objective function for a local minimum because it is non-convex and thus does not necessarily have a unique minimum [11].

3 Patch-Based Mesh Optimization

Many optimization algorithms could be applied to solve the above mesh shape-quality optimization problem. One algorithm which has been successfully applied to solve this problem is FeasNewt [11], developed by Munson. The local version of this algorithm corresponds to a block coordinate descent method where each subproblem is solved with an inexact Newton method. The global version performs an inexact Newton method with an Armijo linesearch. For a detailed comparison of the local and global versions of FeasNewt see [12].

We focus on the intermediate case and propose solving the mesh quality improvement problem using a patch-based approach by iteratively optimizing vertices in each patch of the mesh in a method inspired by the Schwarz domain decomposition methods [16]. The patches are defined by assigning vertices to patches and then adding in additional adjacent vertices according to the overlap calculation (see Fig. 1). For each patch, we solve an optimization problem similar to (1a) through (1c) on the interior and boundary vertices of the patch using a modified version of FeasNewt.

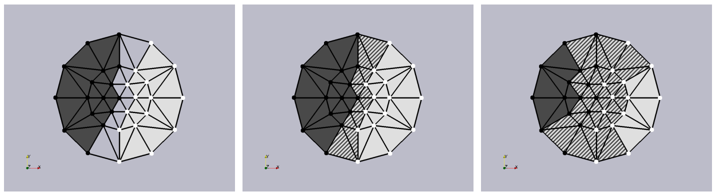

Fig. 1. *Patches with overlap level* 0 *(left),* 1 *(middle), and* 2 *(right). Two patches are shown by the light and dark gray regions, with the initial vertex assignment given by the vertex colors. The striped region denotes elements that are assigned to both patches.*

An outline of the major steps in our patch-based mesh optimization algorithm is given below. The first step is to partition the mesh M into n patches (currently done using Metis [17]). Second, the overlap region for each patch is computed. In particular, all elements adjacent to the current patch are added for each level of overlap. Third, until convergence is obtained, global mesh vertex repositioning is performed on each patch, and the new vertex positions in the overlap regions are communicated to overlapping patches. FeasNewt [11] is used to perform the vertex repositioning step. Code development is being performed in Mesquite [18], the Mesh Quality Improvement Toolkit, since it is extensible and contains both local and global implementations of FeasNewt. We added a new class, MetisPatch, to Mesquite so that FeasNewt could be used to optimize vertices in various sized patches.

Patch-Based Mesh Optimization Algorithm

```
Input:  M = mesh; n = number of patches desired,
  l = level of overlap desired.
1.  Partition M into n patches.
2.  Compute overlap of level l.
3.  While (!converged) do
4.    For each patch
5.      Perform global mesh optimization on the patch.
6.      Communicate new vertex positions to overlapping patches.
7.    End for
8.  End while.
```

4 Numerical Experiments

Numerical experiments were designed to investigate the effects of the number of patches and the amount of patch overlap on mesh optimization time. All experiments were run to a high level of mesh quality, selected so that subsequent iterations provided little improvement. Results may differ if meshes of lesser quality are acceptable. Results from the patch-based mesh optimization experiments were compared with those obtained from global and local mesh optimization using FeasNewt. CUBIT [19] was used to generate meshes for the duct and gear geometries (see Fig. 2) for the numerical experiments. The experiments were performed on a Dell machine with an Intel Core 2 Duo processor (2.13 GHz, 4 MB L2 cache), 3 GB of RAM, and running Linux.

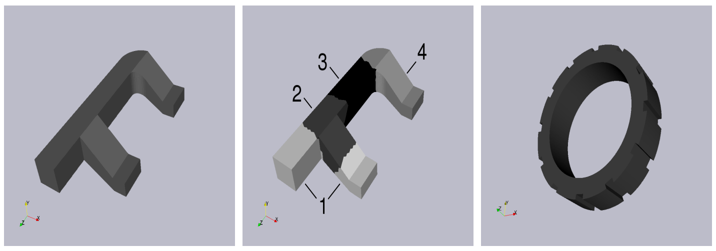

Fig. 2. *The duct geometry (left), large duct mesh decomposed into patches (middle), and gear geometry (right).*

For the first experiment, a small duct mesh was created having 13,193 vertices and 65,574 tetrahedral elements. This mesh was divided into 2, 4, 6, 8, 16, 32, and 64 patches. Mesh optimization was performed with 1, 2, 3, 4, 8, and 16 levels of overlap and was terminated when the average mesh quality reached 1.1401785. (Figures for this experiment have been omitted due to space constraints.) For this mesh, the best patch-based result required 1.67 seconds and was obtained with 2 patches and overlap level 2. The greatest time required was 8.99 seconds and was obtained for 64 patches and overlap level 16. In this case, the best results were obtained for a smaller number of patches and level of overlap. At some point, decomposing the mesh into more patches or increasing the amount of overlap does not pay off. For this mesh, we outperformed local mesh optimization (29.5 seconds) for every combination of (patches, overlap level); however, we were not able to outperform global mesh optimization (1.55 seconds). Larger meshes are required before our patch-based mesh optimization algorithm outperforms the global algorithm. Decomposing the mesh into overlapping patches was shown not to effect the mesh quality. In all cases (global, local, patch-based), the mesh optimization algorithms converged to meshes of similar quality.

For the second experiment, a large duct mesh was generated having 177,887 vertices and 965,759 tetrahedral elements. To create a more difficult optimization problem, the initial mesh was also randomly perturbed making it further from optimal. The initial quality of the large duct mesh can be found in Table 1. For this experiment, the mesh was divided into 2, 4, 8, 16, 32, 64, and 96 patches. Mesh optimization was performed with 1, 2, 3, 4, 8, 16, and 24 levels of overlap and was terminated when the average mesh quality reached 1.135295. Typical results for varying the number of patches (for a fixed overlap level) and varying the amount of overlap (for a fixed number of patches) are shown in the top row of Fig. 3. The bottom row of Fig. 3 shows the total time versus the number of patches (with a fixed overlap level) and the total time versus the overlap level (with a fixed number of patches) for all combinations of (number of patches, overlap level) using the patch-based mesh optimization algorithm. The solid lines in these plots represent the time taken by global mesh optimization (46.49 seconds, 6 iterations) and local mesh optimization (117.68 seconds, 8 iterations).

Table 1. Initial and final quality of large duct mesh

Mesh	**Minimum**	**Average**	**RMS**	**Maximum**	**Std. Dev.**
Initial	1.00037	1.15198	1.15843	20.6399	0.122132
Final (16 patches, 8 overlap)	1.00037	1.13529	1.14019	11.0809	0.105568

For patch-based mesh optimization, the fastest result (35.11 seconds, 2 iterations, final quality shown in Table 1) was obtained for 16 patches and overlap level 8 (a medium number of patches and overlap level) on the large duct mesh. This result was 1.4% faster than the next best result and outperformed global mesh optimization by 24.5% and local mesh optimization by 70.2% by reducing the number of nonlinear iterations (final quality of the local and global meshes, not included due to space, is very similar to that of the patch-based one). The ob-

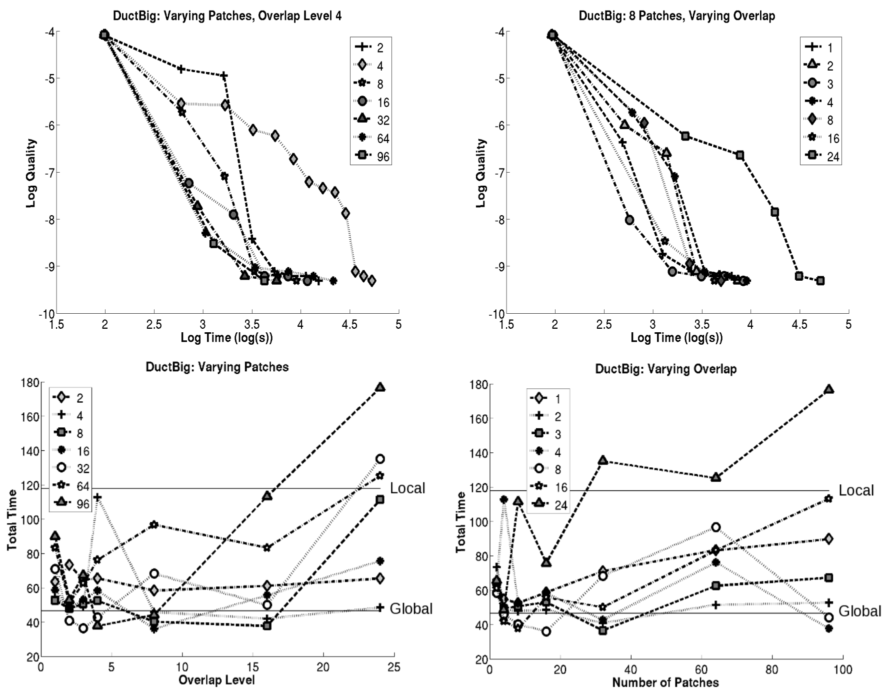

Fig. 3. *Top row: Typical results: Varying patches, overlap level 4 (left) and 8 patches, varying overlap (right) for the ductbig mesh. Bottom row: Total time versus overlap level, varying patches (left) and total time versus patches (right) for the ductbig mesh. global and local times are indicated by lower and upper horizontal lines (respectively).*

served range of mesh optimization times for the patch-based mesh optimization method is 35.11 to 175.32 seconds.

From these figures, it can be seen that nine combinations of (patches, overlap level) outperformed the global mesh optimization method. In addition, all but three combinations outperformed the local mesh optimization method. It should also be noted that the (4 patches, overlap level 4) combination was much slower than others with overlap level 4 as seen in Fig. 3. This is likely since Metis decomposed the large duct mesh into four patches, with one of the patches consisting of two non-contiguous blocks; see Fig. 2. This behavior was not observed on other mesh decompositions. We plan to investigate the use of other mesh partitioners with the goal of improving the mesh decomposition.

To give a better idea of the patch-based mesh optimization performance, Table 2 gives a breakdown of the mesh optimization time into the following categories: mesh decomposition time, overlap calculation time, and vertex repositioning time for three sample performances: slow, medium, and fast (in terms of total time required to solve the problem). Thus, the decomposition time is small and roughly constant; the overlap time is intermediate and variable, and the optimization time is large and variable.

Table 2. Timing on large duct mesh

	Decomposition		Time (s)			
Type	**Patches**	**Overlap**	**Decomposition**	**Overlap**	**Optimization**	**Total**
Slow	64	24	0.910 (0.9%)	13.22 (12.8%)	89.2 (86.3%)	124.16
Medium	2	16	0.840 (2.1%)	7.12 (17.5%)	32.8 (80.5%)	60.29
Fast	16	8	0.830 (3.0%)	7.45 (26.9%)	19.4 (70.1%)	35.11

For the final experiment, a mesh was created on the more complex gear geometry (which contains a hole and a concave exterior boundary). The gear mesh has 102,405 vertices and 544,649 tetrahedral elements; the initial quality of this mesh can be found in Table 3. For this experiment, the mesh was divided into the following numbers of patches: all powers of 2 from 2 to $2^8 = 256$ and the intermediate values halfway between consecutive powers of 2. Mesh optimization was performed with 1, 2, 3, 4, 8, 12, 16, and 24 levels of overlap and was terminated when the average mesh quality reached 1.1203. Figures similar to those in the top row of Fig. 3 have been omitted due to space constraints.

Fig. 4 shows the total time versus the number of patches (for a fixed overlap level) and the total time versus the overlap level (for a fixed number of patches) for all combinations of (patches, overlap level) for the patch-based mesh optimization algorithm. For comparison purposes, the solid lines in this figure represent the time taken by global mesh optimization (52.64 seconds, 10 iterations) and local mesh optimization (227.53 seconds, 25 iterations).

Table 3. Initial and final quality of gear mesh

Mesh	**Minimum**	**Average**	**RMS**	**Maximum**	**Std. Dev.**
Initial	1.00004	1.16299	1.16748	2.28506	0.102356
Final (256 patches, 1 overlap)	1.00016	1.12071	1.12417	4.32766	0.088178

The fastest result (16.17 seconds, 3 iterations, final quality shown in Table 3) was obtained for 256 patches and overlap level 1, corresponding to an increased number of patches and small overlap level. This was 0.3% better then the next fastest result and outperformed global mesh optimization by 69.3% and local mesh optimization by 92.9%. Numerous combinations of (patches, overlap level) outperformed the global mesh optimization method, and all but one outperformed the local mesh optimization method. The range of mesh optimization times for the patch-based method was 16.17 to 268.12 seconds.

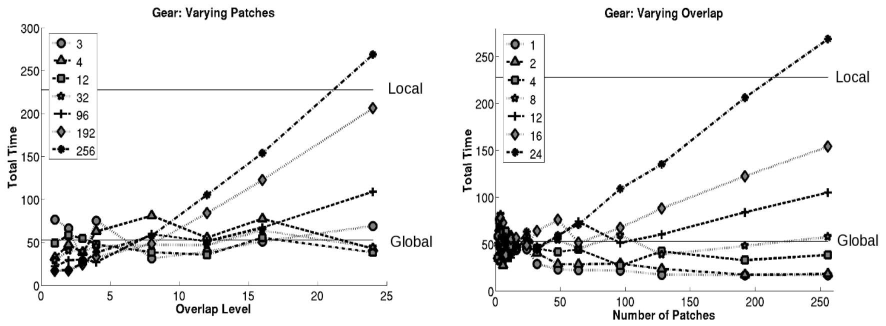

Fig. 4. *Total time vs. overlap level, varying patches (left) and total time vs. patches (right) for the gear mesh. (Some intermediate results are not shown for figure clarity). Global and local times are indicated by lower and upper horizontal lines, respectively.*

5 Conclusions

We proposed a patch-based optimization algorithm for mesh quality improvement. The algorithm decomposes the mesh into patches, computes the desired level of overlap, and optimizes vertices in the overlapping patches. Numerical experiments showed that patch-based mesh optimization with either an intermediate number of patches and intermediate level of overlap or an intermediate number of patches and small level of overlap give best results for larger meshes (depending on mesh geometry and initial mesh quality). For the small mesh, a small number of patches and small overlap level yielded the best result. For this mesh, our patch-based algorithm outperformed the local algorithm, but did not outperform the global mesh optimization algorithm. For large meshes, the patch-based algorithm outperformed both the global algorithm with several combinations of (patches, overlap level) and the local algorithm with nearly all combinations due to a reduced number of nonlinear iterations. These results suggest a parallel version of our algorithm will likely be competitive with existing parallel local and global mesh optimization algorithms (e.g., [20]).

We have identified three avenues for future work. First, we will study the effect the mesh partitioner has on the algorithm by considering other partitioners such as geometric separators [21]. Second, we plan to develop a theory of patch-based mesh optimization by connecting to domain decomposition methods such as the multiplicative Schwarz methods. Third, we will parallelize our algorithm to enable optimization of large-scale meshes. Associated challenges include: distributing the patches, handling overlaps, finding optimal patch ordering within processors, and determining the optimal communication patttern.

References

1. Babuska, I., Suri, M.: The p and h-p Versions of the Finite Element Method, Basic Principles, and Properties. SIAM Rev. 36, 579–632 (1994)

2. Berzins, M.: Solution-based Mesh Quality for Triangular and Tetrahedral Meshes. In: 6th International Meshing Roundtable, pp. 427–436. Sandia National Laboratories, Albuquerque, NM (1997)
3. Berzins, M.: Mesh Quality - Geometry, Error Estimates, or Both? In: 7th International Meshing Roundtable, pp. 229-237. Sandia National Laboratories, Albuquerque, NM (1998)
4. Bank, R., Smith, B.: Mesh Smoothing Using a Posteriori Error Estimates. SIAM J. Num. Anal. 34: 979–997 (1997)
5. Montenegro, R., Escobar, J., Montero, G., Rodriguez, E.: Quality Improvement of Surface Triangulations. In: 14th International Meshing Roundtable, pp. 469-484. Sandia National Laboratories, Albuquerque, NM (2005)
6. Branets, L., Carey, G.: Smoothing and Adaptive Redistribution for Grids with Irregular Valence and Hanging Nodes. In: 13th International Meshing Roundtable, pp. 333-344. Sandia National Laboratories, Albuquerque, NM (2004)
7. Freitag, L, Ollivier-Gooch, C.: Tetrahedral Mesh Improvement Using Swapping and Smoothing. Int. J. Numer. Meth. Eng, 40, 3979–4002 (1997)
8. Canann S.A., Stephenson, M.B., Blacker, T.: Optismoothing: An Optimization-Driven Approach to Mesh Smoothing. Fin. Elem. Anal. Des. 13, 185–190 (1993)
9. Parathasarathy V.N., Kodiyalam S.: A Constrained Optimization Approach to Finite Element Mesh Smoothing. Fin. Elem. Anal. Des. 9, 309–320 (1991)
10. Zavattieri P., Dari E., Buscaglia G.: Optimization Strategies in Unstructured Mesh Generation. Int. J. Numer. Meth. Eng. 39, 2055–2071 (1996)
11. Munson, T.: Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric. Math. Program., 110, 561–590 (2007)
12. Freitag Diachin L, Knupp P., Munson T., Shontz S.: A Comparison of Two Optimization Methods for Mesh Quality Improvement. Eng. Comput. 22, 61–74 (2006)
13. Liu, A., Joe, B.: Relationship Between Tetrahedron Quality Measures. BIT., 34, 268–287 (1994).
14. Munson, T.S.: Mesh Shape-Quality Optimization Using the Inverse Mean Ratio Metric: Tetrahedral Proofs. Technical Memorandum ANL/MCS-TM-275, Argonne National Laboratory, Argonne (2004)
15. Brenner, S.C., Scott, L.R.: The Mathematical Theory of Finite Element Methods. Springer, Berlin Heidelberg, New York (2002)
16. Smith, B., Bjorstad, P., Gropp, W.: Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press (1996)
17. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. In: International Conference on Parallel Processing, pp. 113-122. (1995)
18. Brewer, M., Freitag Diachin, L., Knupp, P., Leurent, T., Melander, D: The Mesquite Mesh Quality Improvement Toolkit. In: 12th International Meshing Roundtable, pp. 239–250. Sandia National Laboratories, Albuquerque, NM (2003)
19. Sandia National Laboratories: CUBIT: Geometry and Meshing Toolkit. http://www.cubit.sandia.gov.
20. Freitag, L., Jones, M., Plassmann, P.: A Parallel Algorithm for Mesh Smoothing. SIAM J. Sci. Stat. Comp. 20, 2023–2040, (1999)
21. Miller, G., Teng, S., Thurston, W., Vavasis, S.: Geometric Separators for Finite-Element Meshes. SIAM J. Sci. Comp. 19, 364–386 (1998)