The Effect of Vertex Reordering on 2D Local Mesh Optimization Efficiency

Suzanne M. Shontz[1,*] and Patrick Knupp[2,**]

[1] Department of Computer Science and Engineering, 343J Information Sciences and Technology Building, The Pennsylvania State University, University Park, PA 16802
shontz@cse.psu.edu

[2] Applied Mathematics and Applications, MS 1318, P.O. Box 5800, Sandia National Laboratories, Albuquerque, NM 87185-1318
pknupp@sandia.gov

1 Local Mesh Optimization

Many applications in computational science such as heat transfer, advection-diffusion, and fluid dynamics numerically solve partial differential equations. To numerically solve the equations, finite element, finite volume, and other PDE-discretization methods are commonly used, along with meshes to discretize the physical domain. It is well-known that the mesh and its quality can greatly impact the accuracy of simulations, as well as solver efficiency [1], [2]. Mesh quality can be improved by various methods including adaptivity [3], [4], smoothing [5], [6], and swapping [7], [8]. In mesh smoothing, one employs vertex-movement strategies to change the coordinates of mesh vertices, leaving initial mesh connectivity intact. Mathematically rigorous methods for formulating the smoothing problem entail the use of an objective function that measures quality; included in this category are the variational methods for structured meshes [9], [10], [11], and the direct optimization methods based on mesh entities such as lengths and angles [12], [13].

A commonly-used method for smoothing meshes is the *local patch* method wherein one loops over a set of patches containing one free vertex and the

* This work was funded in part by NSF grant CNS 0720749 and a Grace Woodward grant from The Pennsylvania State University.

** This work was funded by the Department of Energy's Mathematics, Information, and Computational Science Program (SC-31) and was performed at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. Department of Energy under contract DE-ACO4-94AL85000. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

adjacent vertices [14], [15], [16]. On each patch, the coordinates of the free vertex are updated by using an update rule or by solving a local optimization problem. These solution methods are required to be very fast in order to keep pace with other parts of a PDE-based simulation. As such, one usually only performs a few sweeps to obtain an inexact solution, i.e., one that is closer to optimal than the initial mesh, but differs from the true optimum.

The local patch smoothing method strongly resembles local iterative methods such as Gauss-Seidel for solving systems of equations. From that perspective, it is natural to ask whether or not the knowledge of linear solvers can be applied to mesh smoothing methods. One solver topic that seems particularly relevant to local mesh smoothing is that of vertex ordering. It is well-known that, in solving systems of equations via local relaxation methods, the order in which the local sets of equations associated with a vertex are considered can make a difference in the time needed to solve the equations. For example, [17] describes natural and red-black orderings, also known as point partitionings. This must also be the case when smoothing meshes via local patches; however, it does not appear to have been investigated to any large extent. In the present study, we examine the effect that altering the vertex ordering has on the CPU time to solution, with the goal of accelerating the convergence.

2 Timing Considerations

Pseudocode for the local patch optimization scheme is shown below. The smoothing or optimization procedure used to update the free vertex coordinates may itself be iterative, but the details of this inner iteration procedure are not needed for the present purposes. For this study, only interior vertices are allowed to move; boundary vertices are treated as fixed and therefore do not appear in the smooth list. Note that if the termination criterion (for the outer 'while' loop) is not tight, then the final mesh will depend on which fixed vertex ordering is selected.[1] The outer termination criterion used in this study is a comparison of the value of the objective function at each iteration (representing the mesh quality) to a predetermined value. Therefore, we are interested in the ordering scheme that reaches the given level of quality in the least amount of time. The inner termination criteria varied as described later.

To assess the effectiveness of the reordering schemes in different situations, we recorded the Optimization Time per Iteration (OTI), which is the difference between Timer3 and Timer2. The cumulative optimization time (COT), T_j, is the sum of all the OTI's over the 'while' loop, with j denoting the reordering scheme $j = 1, 2, ..., J$.[2] Also recorded were the cumulative reordering time, CRT, (Timer4 - Timer3) and the total time, TT, (Timer 5 - Timer 1).

[1] In general, the final mesh can also depend on which fixed vertex ordering is selected even when the tolerance is tight due to the existence of non-unique minimae. We have attempted to avoid this possibility by using a convex local metric.

[2] The timings include a negligible amount of time due to overhead operations.

```
Pseudocode for Local Mesh Smoothing and Optimization

1. Initialization: vertex list, termination criterion,
     tolerances, objective function, control parameters.
   -- CPU Timer_1
2. While tolerance not satisfied and iteration count
     not exceeded
          -- CPU Timer_2
          Loop over list of free vertices
            - update free vertex coordinates (optimization)
          End loop over free vertices
          -- CPU Timer_3
          Reorder vertex list every k-th iteration
          -- CPU Timer_4
   End while loop
   -- CPU Timer_5
```

3 Study Design

There are many variables, parameters, and choices one might make in a study of the impact of vertex reordering on the overall efficiency of local patch-based mesh optimization. Among the possible choices are different quality metrics, objective function templates, optimization solvers, serial or parallel solvers, computer architecture, 2D/3D, element type, type of reordering scheme, mesh anisotropy/isotropy, mesh heterogeneity/homogeneity, mesh connectivity, tightness of the termination criterion of the outer 'while' loop, and the number of free vertices in the mesh. For this initial work we confine our focus by fixing the local quality metric ($||T - I||_F^2$), the template (linear averaging), the solver (Feasible Newton [18]), serial computation, and the dimension (2D). The quality metric is described in [19] where its utility is demonstrated. The quality metric takes on values from 0 to ∞ with 0 being the best value; a value of 0 is obtained when $A = W$, i.e., whenever the physical and reference elements agree. Further, the metric and objective function are proven convex in [20]. The machine employed for this study is equipped with two Intel (R) Xeon (TM) processors each having four cores; each processor has an 8MB cache shared between its cores. The 64-bit machine has 4GB of RAM and runs Linux. Optimizations and reorderings were performed using the Mesquite code (Version 1.99 Alpha) [21].

The remaining free variables that were studied in this research are: (i) several vertex reordering schemes to be described in the next section, (ii) the number of free vertices in the optimization problem, (iii) mesh type (element type and degree of heterogeneity/isotropy), and (iv) the frequency with which the vertex list reordering is performed.

The outer termination criterion for the runs was based on the desire to have them complete in roughly five minutes, because, in this amount of time, the resulting meshes are neither highly accurate nor grossly inaccurate (the inaccurate

case, i.e., requiring about one minute, was also considered). The outer termination criterion value was determined in a set of baseline calculations by running the code for more than five minutes to find the value of the objective function at five minutes. This value was then used to terminate subsequent runs.

The optimization problem that is solved in this study used an objective function that was the average over the entire mesh of the values of the local metric at the element corners (quadrilateral case) or at the barycenter (triangle case). The matrix $T = AW^{-1}$ was determined from A and W, where A represents the quality of the physical mesh, and W represents the target quality, as determined from a reference mesh. See [22] and [19] for details.

4 Vertex Reordering Schemes

Vertex reordering can be done either statically or dynamically. In the static case, one takes as input the initial ordering of the vertex list provided to the optimization scheme and reorders using some criterion. The reordered list is then used and kept fixed during the optimization. In the dynamic case, the reordered list is updated at the end of each iteration (or at the end of every k^{th} iteration) within the optimization, thus creating a sequence of lists. Both strategies were considered in this study. The reordered lists contain the same number of vertices as the initial list. The challenge in all of these approaches is to devise a reordering scheme that accelerates convergence while keeping the computational cost of the reordering low so that the latter does not dominate the time to optimize.

The vertex reordering schemes chosen for this study were ones that were easy to implement and make sense from a geometric point of view. Admittedly, from a theoretical point of view, it is more appealing to choose reordering schemes based on prior knowledge and experience from linear solvers. However, since it is not necessary to derive the global linear systems in order to solve our optimization problem, this approach is reserved for future work. Specific vertex reordering schemes that were investigated are described next.

Scheme (N): Null Order. Do not reorder the vertex list.
Scheme (R): Random Order. Generate a random integer from 1 to N, with N being the total number of local patches in the mesh. Place the vertex assigned the value 1 first in the list and N last.
Scheme (WQP): Order by Worst Quality Patch. Evaluate the objective function on a local patch to measure local patch quality. Sort by putting the worst quality patch first in the list, and so on.
Scheme (GAVM): Order by Greatest Absolute Vertex Movement. Evaluate the position of the free vertex in the local patch before and after the optimization and calculate the absolute distance moved. Sort by putting the patch with the greatest absolute distance moved first in the list, and so on.
Scheme (GRVM): Order by Greatest Relative Vertex Movement. Evaluate the relative distance moved by the free vertex in the local patch before and

after the local optimization. Relative distance was measured by dividing the absolute distance moved by the initial absolute distance moved; the normalizing quantity thus varies from patch to patch. Sort by putting the patch with the greatest relative distance moved first in the list, and so on.
Scheme (LNG): Order by Largest Norm of Gradient. Evaluate the ℓ_2-norm of the local gradient of the objective function. Sort by putting the largest norm first in the list, and so on.

For the schemes WQP, GAVM, GRVM, and LNG, there were associated 'distance' schemes D-WQP, D-GAVM, D-GRVM, and D-LNG which are created by measuring the distance of the free vertex in each patch from the position of the free vertex which had the (i) worst patch quality, (ii) greatest average vertex movement, (iii) greatest relative vertex movement, or (iv) largest norm of the gradient, respectively. Then sort by putting the patch with the smallest distance first in the list, and so on.

For each of the schemes above, there is a corresponding ordering scheme produced by reversing the ordering (the abbreviation for these schemes is the same as the forward ordering except there is a 'minus' at the start of the abbreviation). For example, scheme -WQP orders the vertex patches from best quality first to worst quality last. There are a total of 20 ordering schemes.

A QuickSort algorithm was used in this study to do the reordering. The expected runtime of the sort is $O(N \log N)$, with the worst case being $O(N^2)$, where N is the number of patches to smooth per outer iteration. The larger the number of patches, the more time it takes to reorder them, so the key question is whether or not the time saved by the reordering (as a function of N) increases as fast as the time to do the reordering.

5 Mesh Selection Criteria

Five meshes were created, as described in Table 1. Two basic cases are the structured quadrilateral meshes and the unstructured triangle meshes. The meshes all have the same physical domain, i.e., the horseshoe. The five mesh types contain roughly 20K, 40K, or 80K free vertices each, for a total of 15 meshes included in the study.

Table 1. Mesh Identification Table

MeshID	Description
SQI	Structured Quad, 1:1 cell aspect ratio
SQII	Structured Quad, 1:5 cell aspect ratio
SQIV	Structured Quad, Biased 1:5 cell aspect ratio
UTI	Unstructured Triangle Mesh, Trapezoidal Logical Domain
UTIV	Unstructured Triangle Mesh, Biasing

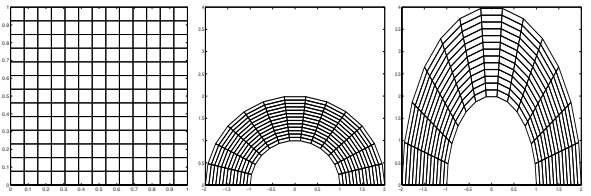

Fig. 1. Mesh Type SQI for the Unit Square, Reference Semi-Annulus, and Horseshoe

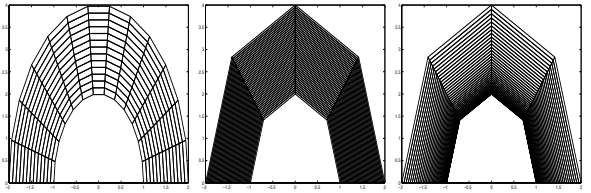

Fig. 2. The Three Quadrilateral Mesh Types. Left to Right: SQI, SQII, SQIV

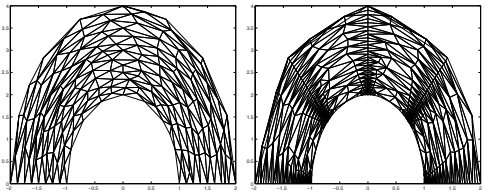

Fig. 3. Two Triangular Mesh Types. Left to Right: UTI, UTIV

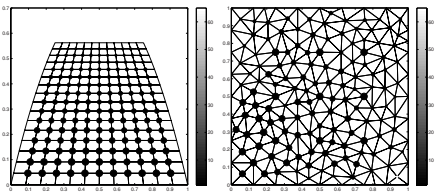

Fig. 4. Initial Vertex Ordering on Structured and Unstructured Meshes

The approach to generating these meshes was to first generate regular meshes on the unit square. For each such mesh, a reference mesh on the semi-annulus and an initial mesh on the horseshoe were created by analytically mapping each vertex in the unit mesh to give corresponding coordinates on the other two domains. An example of the mesh triples (logical, reference, and horseshoe domains) generated using the mappings for Mesh Type SQI is given in Figure 1. Optimization

was always performed on the horseshoe domain using target matrices based on the corresponding semi-annulus mesh.

Figures 2 and 3 show coarse versions of the Horseshoe meshes in Table 1.

Figure 4 shows the initial ordering of the vertices as given by the mesh generator in the structured and unstructured meshes on the unit square. The mapping, of course, preserves the orderings. Large vertices occur early in the list, while smaller vertices occur towards the end. The initial (row-by-row) ordering for all of the structured meshes is the same as shown in the figure, while the ordering for the unstructured mesh in the figure is a representative ordering for both types of unstructured meshes used in this document.

6 Numerical Experiments and Results

Eight numerical experiments were run using the Mesquite code to investigate the behavior of the reordering schemes under various circumstances such as mesh type, mesh size, and reordering strategy. Table 2 summarizes the eight experimental set-ups.

Table 2. Table of Experiments

Exp. No.	Exp. Name	Mesh Types	Mesh Sizes	Schemes	S/D	Frequency	# Runs
6.1	Baseline	all 5	all sizes	N	D	N/A	30
6.2	Static Reordering	all 5	80K	20	S	N/A	100
6.3	Sensitivity to IM	SQI	80K	20	S/D[5]	N/A / 1[6]	40
6.4	Dynamic Reordering	all 5	all sizes	20	D	1	300
6.5	Frequency	all 5	80K	20	D	1,2,5,10	400
6.6	ITC	all 5	80K	20	D	1	100
6.7	Inaccurate	all 5	all sizes	20	D	1	300
6.8	Red-Black	all SQ	80K	20	D	1	60

[5] One static (S) and one dynamic (D) experiment were conducted.
[6] The frequency value is not applicable (N/A) for the static experiment and is set to one (1) for the dynamic experiment.

The output of these experiments took the following form: 1) input summary tables to record the key inputs, 2) overlay plots showing average mesh quality vs. total time (TT) for all twenty reordering schemes, and 3) ranking tables giving optimization (COT), reordering (RT), and total times (TT), along with rankings from 1 to 20 for each of these items (plus a ranking based on outer iteration count). Typical examples of the quality vs. time plots are shown in Figure 5. Additional post-processing of the results was also performed. The purpose of each experiment is described next, along with selected results. At the end of the section, observations that pertain to all of the experiments are given.

6.1 Baseline Experiment

The purpose of this 'experiment' was to determine the quality level that would result in 1 or 5 minutes of smoothing. One minute of smoothing gives a relatively inaccurate approximation to the optimal mesh, while five minutes gives moderately accurate results. Runs were terminated after one or five minutes, and the quality level was recorded, to be used in subsequent experiments.

Selected Results
By examining the quality vs. time plots, it was noted that the shape of these curves generally followed two basic patterns: either the curves were straight lines, or they were highly curved, similar to a rapidly decaying L-shaped function (see Figure 5). The pattern was found to continue over the seven other experiments that were performed. The explanation appears to be that if the initial mesh is relatively 'far' from the optimal mesh then rapid convergence occurs at early times and linear convergence at later times, giving the L-shaped curve. On the other hand, if the initial mesh is relatively 'close' to the optimal mesh, one only sees the tail of the L-shaped function, which appears linear. For the most part, the curves for the SQI and SQII meshes appear linear, while the SQIV, UTI, and UTIV meshes often showed L-shaped convergence behavior. In general, the curves for the finer meshes (40-80K) had a steeper slope than the coarser meshes (20-40K) and satisfied the convergence criterion more quickly.

6.2 Static Experiment

The purpose of this experiment was to observe the behavior of the reordering schemes using a 'static' reordering strategy in which the reordering is done only once, i.e., before optimization begins.

Selected Results
The average quality vs. time curves were observed to be monotonically decreasing in most cases; however, for a few ordering schemes (GRVM, R, WQP) on the SQII meshes, they actually increased initially, before decreasing and satisfying the convergence tolerance.[3] This placed these schemes in last place in terms of TT ranking. In very rare instances (and this applies to all experiments), quality vs. time curves were observed to cross one another, meaning that ordering schemes that were effective at early times, became relatively ineffective at later times. For the most part, however, quality vs. time curves do not cross one another. Moreover, in the static experiment, they tend to be parallel to one another, as seen in Figure 5.

The choice of reordering scheme had a noticeable effect on time to convergence (TT). The variation in total time-to-convergence relative to the average time-to-convergence for the given mesh type over the reordering schemes in the static

[3] The increase is attributed to the fact that it is theoretically possible for the local average quality to increase when using the Feasible Newton solver on a local patch; a poor choice of search direction is one way this can happen. If the local average quality can increase, then so can the global average quality.

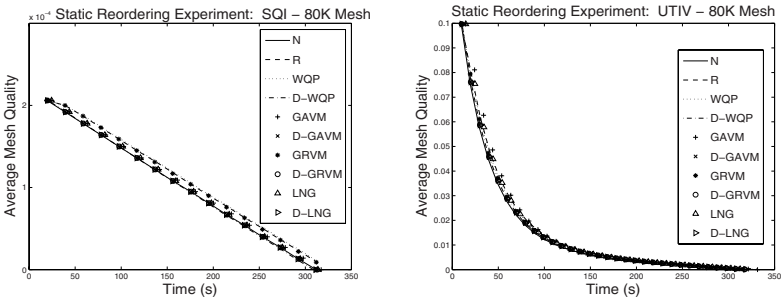

Fig. 5. Sample Quality vs. Time Plots: Linear (left) and L-shaped (right)

experiments was, for the 80K meshes, SQI: 312 ± 5 seconds, SQII: 366 ± 61 seconds, SQIV: 258 ± 62 seconds, UTI: 312 ± 10 seconds, and UTIV: 319 ± 13 seconds. That is, the variation in time-to-convergence due to the choice of reordering schemes was: 1.6% for SQI, 16.7% for SQII, 24.0% for SQIV, 3.2% for UTI, and 4.1% for UTIV. It appears that the choice of reordering scheme is more important for biased or anisotropic mesh types. Although the difference between using one reordering scheme versus another is noticeable, the maximum effect was less than 40% of the total run times.

Nearly all reorderings took less than 1.5 seconds. Smoothing times varied from 105 to 315 seconds, so reordering time was less than one percent of the smoothing time. In terms of total time, -N was the best at 196 seconds, while -GAVM was the worst at 320 seconds. Notably, Scheme N took about 314 seconds while scheme -N took 196 seconds, about a 30% difference in total time. Other mesh types had different reordering schemes giving the least TT.

6.3 Sensitivity of Results to Initial Mesh

The purpose of this experiment was to observe the behavior of the reordering schemes when a different initial mesh was used, with both static and dynamic (see Experiment 6.4) reordering strategies. The second initial mesh differed from the first initial mesh only in the vertex coordinates. The vertices in both of these initial mesh types were ordered in the same way (see Experiment 6.8 for an example of different vertex orderings in the initial guess). The vertex coordinates in the second initial mesh were obtained from the first initial mesh by randomly perturbing them by the largest possible amount without creating inverted mesh elements; the second mesh is farther from optimal.

Selected Results for the Static Case

The quality vs. time plots showed that the curves that were linear when using the first initial guess became L-shaped when using the second initial guess, as expected since the second initial guess is farther from optimal than the first.

The variation in total time-to-convergence relative to the average time-to-convergence for the 80K SQI mesh type over the reordering schemes in this

experiment was 895 ± 51 seconds, or 5.6%, compared to 1.6% when the first initial guess was used. Thus, it appears that the farther the initial guess is from optimal, the larger the variation, and thus the more significant the choice of reordering scheme becomes.

6.4 Dynamic Experiment

The purpose of this experiment was to observe the behavior of the reordering schemes using a different reordering strategy. The 'dynamic' strategy is to perform the reordering once per outer iteration. This will cost more than the static strategy in terms of time spent reordering, so the question is whether or not the added expense is worth it in terms of the total CPU time expended.

Selected Results
The quality vs. time curves differed from those in the static case in that the linear curves that were parallel in the static case remained linear, but with differing slopes, depending on the reordering scheme. This reflects the fact that in the dynamic strategies, differences amongst ordering schemes become cumulative with time, whereas this does not happen in the static case. As a result, the quality vs. time curves in the dynamic case diverge from one another with increasing time. A similar effect was seen for the L-shaped curves.

The choice of reordering scheme did have a noticeable effect on the time to convergence. The variation in total time-to-convergence relative to the average time-to-convergence for the given mesh type over the reordering schemes in the dynamic experiments was, for the 80K meshes, SQI: 365 ± 55 seconds, SQII: 411 ± 107 seconds, SQIV: 345 ± 116 seconds, UTI: 488 ± 182 seconds, and UTIV: 421 ± 120 seconds. That is, the variation in time-to-convergence due to the choice of reordering schemes was: 15.1% for SQI, 26.0% for SQII, 33.6% for SQIV, 37.3% for UTI, and 28.5% for UTIV. Comparing these results to the static strategy, one sees that both the average total time-to-convergence and the variation is larger in the dynamic case.

Although, the *average* time-to-convergence in the dynamic case is larger than in the static case, it does not necessarily mean that there may not be particular reordering schemes that, when used in the dynamic strategy, are competitive with the static strategy. To investigate, we compared the best time-to-solution for the static and dynamic cases for each of the 80K mesh types (excluding the $\pm N$ and $\pm R$ schemes). It appears that the best dynamic schemes (in terms of least TT) are not quite as effective as the same schemes when used statically. The only exception was for SQII, and that is attributed to the fact that in the static runs the D-GAVM quality versus time curve initially increased, causing it to lag the others; this did not happen in the dynamic case. Evidently, updating the ordering once every iteration is too expensive compared to the gain in overall solution time. The next experiment investigates a modified dynamic strategy.

Table 3. Best Dynamic Scheme vs. Corresponding Static Result

Mesh Type	Best Scheme	Static Time	Dynamic Time
SQI	D-GRVM	312.74	314.81
SQII	D-GAVM	326.13	313.40
SQIV	-GRVM	203.35	228.77
UTI	-GRVM	321.21	346.49
UTIV	D-LNG	318.47	333.08

6.5 Frequency of Dynamic Reordering Experiment

The purpose of this experiment was to observe the behavior of the reordering schemes with a modified dynamic reordering strategy. Instead of reordering every iteration, in this experiment, reordering was performed every iteration, every other iteration, every fifth iteration, and every tenth iteration.

Selected Results
The case of the 80K SQIV mesh is instructive. Consider the behavior of scheme -GRVM vs. the reordering frequency (k); this scheme had the smallest TT over the 20 schemes when $k = 1, 2$.

Table 4. Scheme -GRVM vs. Reordering Frequency

Frequency	COT	COT Rank	RT	RT Rank	TT	TT Rank
1	198.07	1	33.48	12	231.55	1
2	224.76	6	9.55	10	234.31	1
5	261.32	6	0.05	5	261.37	6
10	307.20	7	0.02	6	307.22	7
Static	203.33		0.02		203.35	

The fraction of time spent reordering vs. the total time decreases as reordering becomes less frequent. The optimization time is smallest with frequency 1, as is total time. In this case, reordering every iteration is well worth it compared to the other reordering frequencies. However, the total time in the static case is even less. Figure 6 shows TT vs. reordering frequency for schemes -GRVM, -WQP, and N on the 80K, SQIV mesh. Note that the figure shows that it is best to apply the -WQP scheme as infrequently as possible!

This experiment shows that the ideal frequency of reordering depends on the reordering scheme in question. If the reordering scheme is an effective one for the given problem, it should be applied as frequently as possible.

Does the dynamic strategy ever result in a total time that is less than the corresponding static case? In general, the answer is yes, but not significantly. For each of the five mesh types one can find a reordering scheme whose total time when $k = 1$ or $k = 10$ is less than its own static time, but, even then,

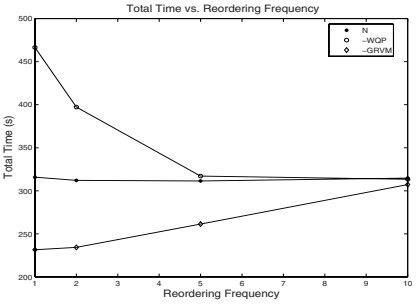

Fig. 6. Total Time vs. Frequency for Schemes -GRVM, -WQP, and N

the dynamic times are only a few percent less than the static times. Therefore, the dynamic strategy, even with infrequent reordering, seems to be mostly less efficient than the static strategy.

6.6 Inner Termination Criterion Experiment

The purpose of this experiment is to determine whether or not the number of inner iterations in the optimization had any significant impact on the results of the reordering experiments.

Selected Results
Increasing the number of inner iterations to 2 increases the time-to-convergence in many of these runs. Since the variation in total times over the twenty reordering schemes does not increase with the number of inner iterations, the impact of the choice of reordering scheme becomes less as the number of inner iterations increases. In addition, the rankings of the reordering schemes changes with the number of iterations.

6.7 Inaccurate Experiment

The purpose of this experiment was to determine how the results of the reordering experiments might change if only an inaccurate (1 minute) solution were sought when dynamic vertex reordering was performed.

Selected Results
The main result here was that, indeed, the rankings of the reordering schemes did change when comparing the 'inaccurate' case to the 'medium-accurate' case. The variations in total time-to-convergence relative to the average time-to-convergence for the given mesh type over the reordering schemes in the Inaccurate Experiment were, for the 80K meshes, SQI: 101 $\pm$ 25 seconds, SQII: 204 $\pm$ 70 seconds, SQIV: 100 $\pm$ 39 seconds, UTI: 113 $\pm$ 35 seconds, and UTIV: 107 $\pm$ 40 seconds. That is, the variation in time-to-convergence due to the choice of reordering schemes was: 24.8% for SQI, 34.3% for SQII, 39.0% for SQIV, 31.0%

for UTI, and 37.4% for UTIV. Comparing these results to the medium-accurate dynamic strategy, one sees that although both the total time-to-convergence and the variation is smaller in the inaccurate case, the choice of reordering scheme becomes more significant percentage-wise.

6.8 Red-Black Experiment

The purpose of this experiment was to investigate whether or not the ordering of the vertices in the initial mesh had a significant impact on the behavior of the ordering schemes. The first (original) initial meshes for the structured mesh types were modified to create a red-black (checkerboard) ordering of the vertices while keeping the vertex coordinates the same. Results were then compared to the original initial mesh.

For this experiment, the percent variation over the reordering schemes was much larger than when using the ordering in the first initial mesh: SQI - 144%, SQII - 209 %, and SQIV - 222 %. For example, the SQI curves diverge much more than in the original ordering, giving a larger variation and much longer run-times overall (300-800 sec vs. 300-450 in original). Scheme rankings are somewhat similar to original, but not too close. For the SQII curves, it was observed that the plots no longer show an initial increase in average quality for any scheme, whereas in the original some did. The scheme rankings are completely different. For SQIV, there was slightly more L-shaped convergence behavior, and overall run times are longer, and the scheme rankings change. The -N scheme is conspicuously bad in the red-black experiments.

The results in the scheme ranking tables showed that for SQI, the best red-black total time is 308.72 sec (for scheme N), while the best original total time was 309.51 sec (again for N), so red-black ordering per se is hardly worse than the original ordering. However, the worst red-black time is 752.21 sec (for GAVM), compared to the worst original (WQP, 420.21 sec). Table 5 directly compares results for three reordering schemes. The table shows that the ordering of the initial mesh itself can alter the final results, but it is hard to predict in advance what will happen.

Table 5. Comparison of Natural and Red-Black Orderings (SQI)

Scheme	COT	COT Rank	RT	RT Rank	TT	TT Rank
D-GRVM-or	309.07	4	5.74	9	314.81	3
D-GRVM-rb	314.57	12	6.48	8	321.05	4
GAVM-or	316.50	14	79.53	16	396.03	17
GAVM-rb	599.84	19	152.37	20	752.21	20
WQP-or	391.84	20	28.37	11	420.21	20
WQP-rb	392.21	15	28.25	10	420.46	16

6.9 Reordering Time

For each of the runs performed in this study, we measured the cumulative CPU time (RT) expended during the reordering step. To investigate which reordering schemes tended to take the least amount of CPU time, we computed a set of histograms. For each reordering scheme we counted the number of times it was ranked first (in terms of reordering time), second, third, and all the way to twentieth. The counting was performed over the entire set of runs performed in this study. Results are summarized in Table 6 and the schemes assigned an overall rank.

Table 6. A Rough Reordering Time Ranking Over All Runs

Rank	Scheme	Mode	2nd Mode	Best Rank	Worst Rank
1	N	1	-	1	1
2	-N	2	3	2	4
3	R	3	4	2	5
4	-R	4	3	2	5
5	GRVM	5	6	4	6
6	D-GAVM	7	8	6	12
7	-GRVM	6	12	4	16
8	D-LNG	7	8	6	11
9	D-GRVM	9	8	6	12
10	WQP	10	11	9	19
11	-WQP	11	14	10	20
12	D-WQP	12	10/13	9	17
13	-D-GRVM	14	11	10	19
14	-D-GAVM	14	13	7	19
15	-D-LNG	15	13	7	19
16	-D-WQP	16	15	13	20
17	-GAVM	17	18	12	19
18	LNG	19	18	10	20
19	GAVM	20	18	10	20
20	-LNG	20	19	11	20

Scheme N ranks first since it takes no CPU time at all, being a null operation. The next three schemes also take very little work since they only reverse a given ordering that was fast to create in the first place. It is difficult to explain the rankings for the remaining schemes, although perhaps a look at operation counts might help. 'Distance' scheme ranks are mixed with the schemes upon which they are based, a somewhat surprising result. As expected, the reverse orderings ranked below the forward orderings (except GAVM).

6.10 Overall Scheme Rankings

The selected results reported in the previous experiments are supplemented by taking a look at TT scheme ranking results over the entire set of runs that were

performed over all the experiments. From the results obtained, it was found that the 'best' orderings varied from experiment to experiment, mesh type to mesh type, mesh size to mesh size, etc., with no discernible pattern. If one were going to do a lot of mesh optimization calculations with these variables fixed, one might benefit by determining which ordering scheme is best in that specific instance. For a general purpose tool such as Mesquite, it is reasonable to ask if there are reordering schemes which generally performed well over all the runs (and which generally did poorly). To investigate, we created a set of histograms, one for each reordering scheme, in which we counted the number of times the scheme was ranked first, second, third, and all the way up to twentieth. The counting was over the entire set of runs.

Table 7. A Rough Scheme Ranking Over All Runs

Rank	Scheme	Mode	2nd Mode	Best Rank	Worst Rank
1	N	1	2	1	20
2	-N	1	2	1	20
3	D-GAVM	3	4	1	15
4	D-LNG	4	3	1	15
5	D-GRVM	4	5	2	16
6	D-WQP	6	9	2	17
7	-D-GAVM	7	8	1	17
8	-D-LNG	9	7	1	17
9	-GRVM	2	15	1	18
10	-D-GRVM	8	9	2	17
11	GRVM	16	2	2	20
12	-D-WQP	10	15	1	19
13	-R	18	17	1	20
14	R	17	18	9	20
15	-GAVM	14	11	5	20
16	-LNG	15	12	2	19
17	LNG	13	18	3	19
18	GAVM	20	12	6	20
19	WQP	20	19	3	20
20	-WQP	20	19	4	20

To rank the schemes using these histograms, we looked at the mode for each scheme and the general distribution around the mode. Based on this, the overall rankings are shown in Table 7, with schemes N, -N, D-GAVM, D-LNG among the best and LNG, GAVM, WQP, and -WQP among the worst. The best 'worst rank' over all the schemes was 15th, so no scheme was immune from a circumstance in which it did poorly. The worst 'best rank' was 9th, so nearly all the schemes had at least one circumstance it which it did very well relative to the others. On the whole, though, the top five of these schemes stand out as being good general purpose reordering schemes. It is perhaps not too surprising to see schemes N and $-N$ at the top of the rankings since they incur virtually no cost in

terms of reordering time. Furthermore, there may be an advantage to leaving the original orderings alone since these reflect the order in which the mesh vertices were likely created when the mesh was generated. Not surprising perhaps is the relatively low rankings of the $\pm R$ schemes since they do not take advantage of problem structure. Nevertheless, they ranked higher than six other schemes. As a general rule, the 'distance'-based schemes were ranked higher than the schemes from which they were derived. For example, GAVM ranks 18th while D-GAVM ranks 3rd; this is because reordering for D-GAVM takes less time than that of GAVM, and often the optimization for D-GAVM is more effective than that of GAVM in that the number of outer iterations is reduced thus requiring less time spent on optimization. Counterintuitively, reordering by patch quality is not advantageous. Ordering of patches to maintain or create locality seems to be an important variable in creating an effective reordering scheme.

7 Conclusions and Future Work

The results of this study are somewhat preliminary but a number of important trends are suggested. First, it appears that the static strategy is superior to the dynamic ones because the cost of additional reordering does not result in enough of a speedup in the optimization process. Second, reordering appears most important when 'inaccurate' meshes are considered satisfactory or when the initial mesh is far from optimal. Third, it is difficult to predict in advance which reordering scheme will be best for a given problem. It was noted that optimization of meshes with strong biasing or heterogeneity will likely benefit most from a consideration of vertex ordering. Additionally, reordering may be beneficial if the ordering of the patches in the initial mesh lacks locality, as seen in the red-black experiment. Fourth, although total time to solution is impacted noticeably by vertex ordering, it is usually within a factor of two, as opposed to orders-of-magnitude. This could change if other mesh sizes and types were considered. Fifth, the natural ordering provided by the mesh generator resulted in scheme N being one of the more efficient orderings. Reversing the ordering of the list (forward to reverse) is inexpensive and often improved TT. Sixth, distance-based reordering schemes appear to be more effective than the schemes upon which they are based; once again, this is most likely due to an increase in locality. Seven, no discernible pattern of results was observed related to mesh size. Perhaps much smaller or larger meshes would have resulted in a pattern with respect to this parameter. Some of these observations can perhaps be explained theoretically, as has been done for the linear equation solver case. Nonetheless, it is useful to have empirical data to suggest possible theoretical results and to provide additional insight.

More data was generated in this study than could be presented due to space limitations. Additional post-processing and analysis of the data is planned in the future to better quantify the results, check for over-looked tendencies, and to determine if there are particular circumstances which can be predicted in advance as to which of the reordering schemes should be used. Future work

on this topic might include the creation of better ordering schemes to provide more locality, further investigations of reordering efficiency vs. mesh size, other dynamic strategies, and the effect of using other optimization solvers.

Acknowledgements

The authors would like to thank Jason Kraftcheck for advice on implementing algorithms in Mesquite, Anirban Chatterjee for C++ consultation, and the reviewers for helpful comments.

References

1. Shewchuk, J.: What is a good linear finite element? (unpublished preprint, 2002)
2. Batdorf, M., et al.: Computational study of the effect of unstructured mesh quality on solution efficiency. In: Proceedings 13th Annual AIAA Computational Fluid Dynamics Conference. AIAA (1997)
3. Bank, R., Smith, R.: Mesh smoothing using a posteriori error estimates. SIAM J. Numer. Anal. 34, 979–997 (1997)
4. Lague, J., Hecht, F.: Optimal Mesh for P1 Interpolation in H1 Seminorm. In: Proceedings 15th Intl. Meshing Roundtable, pp. 259–270. Springer, Heidelberg (2006)
5. Montenegro, R., Escobar, J., Montero, G., Rodriguez, E.: Quality Improvement of Surface Triangulations. In: Proceedings 14th Intl. Meshing Roundtable, pp. 469–484. Springer, Heidelberg (2005)
6. Branets, L., Carey, G.: Smoothing and adaptive redistribution for grids with irregular valence and hanging nodes. In: Proceedings 13th Intl. Meshing Roundtable, pp. 333–344 (2004)
7. Freitag, L., Ollivier-Gooch, C.: Tetrahedral Mesh Improvement Using Swapping and Smoothing. Int. J. Num. Meth. Engr. 40, 3979–4002 (1997)
8. Klingner, B., Shewchuk, J.: Aggressive Tetrahedral Mesh Improvement. In: Proceedings 16th Intl. Meshing Roundtable, pp. 3–23. Springer, Heidelberg (2008)
9. Brackbill, J., Saltzman, J.: Adaptive Zoning for singular problems in two dimensions. J. Comp. Phys. 46, 342–368 (1982)
10. Knupp, P.: Jacobian-weighted Elliptic Grid Generation. SIAM J. Sci. Comp. 17(6), 1475–1490 (1996)
11. Liseikin, V.: A Computational Differential Geometry Approach to Grid Generation. Springer, Heidelberg (2004)
12. Castillo, J.: A discrete variational method. SIAM J. Sci. Stat. Comput. 12(2), 454–468 (1991)
13. Freitag, L., Knupp, P.: Tetrahedral mesh improvement via optimization of the element condition number. Int. J. Num. Meth. Engr. 53(6), 1377–1391 (2002)
14. Canann, S., Tristano, J., Staten, M.: An approach to combined Laplacian & Optimization-Based Smoothing for Triangular, Quadrilateral, & Quad-dominant Meshes. In: 7th Intl. Meshing Roundtable, Dearborn MI, pp. 479–494 (1998)
15. Freitag, L.: On Combining Laplacian and Optimization-Based Smoothing Techniques, Trends in Mesh Generation, AMD vol. 220, pp. 37–43. ASME (1997)
16. Hansbo, P.: Generalized Laplacian smoothing of unstructured grids. Com. Num. Meth. Engr. 11(5), 455–464 (1995)

17. Hageman, L., Young, D.: Applied Iterative Methods. Academic Press, London (1981)
18. Munson, T.: Mesh Shape-Quality Optimization Using the Inverse Mean-Ratio Metric. Math. Program. 110, 561–590 (2007)
19. Knupp, P.: Updating Meshes on Deforming Domains. Communications in Numerical Methods in Engineering 24, 467–476 (2008)
20. Vander Zee, E., Knupp, P.: Convexity of Mesh Optimization Metrics using a Target-Matrix Paradigm, Sandia Laboratories, SAND2006-4975J (2006)
21. Brewer, M., Freitag Diachin, L., Knupp, P., Leurent, T., Melander, D.: The Mesquite Mesh Quality Improvement Toolkit. In: Proceedings 12th Intl. Meshing Roundtable, Sandia Laboratories, Albuquerque, NM, pp. 239–250 (2003)
22. Knupp, P.: Formulation of a Target-Matrix Paradigm for Mesh Optimization, SAND2006-2730J, Sandia National Laboratories, Albuquerque NM (2006)