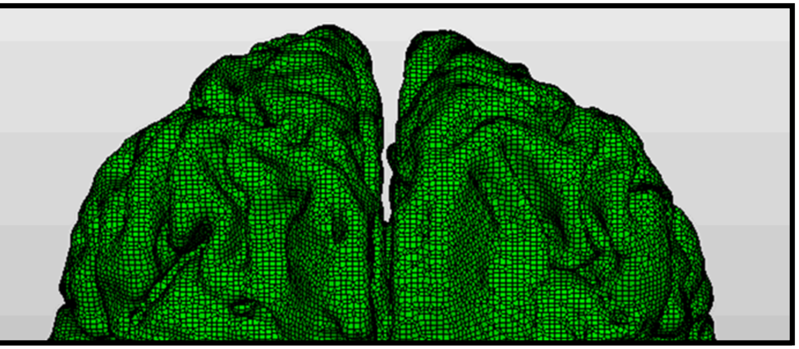
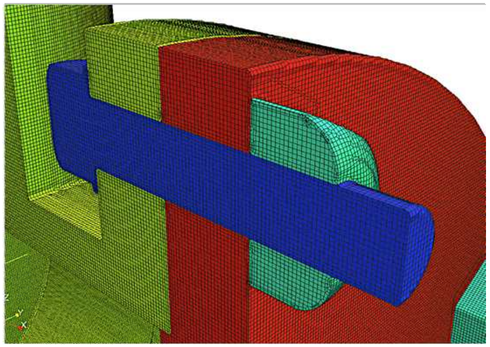
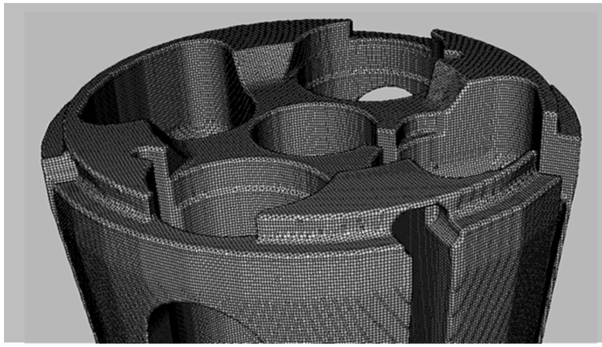
SAND2015-6194C

Exceptional service in the national interest

Sandia National Laboratories







Parallel Mesh Optimization for Grid-based Hex Meshes

US National Congress on Computational Mechanics

San Diego, CA

July, 26th 2015

Steven J. Owen

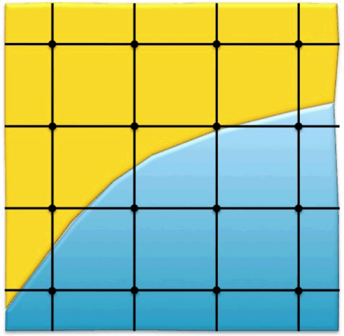
Sandia National Laboratories,

Albuquerque, NM

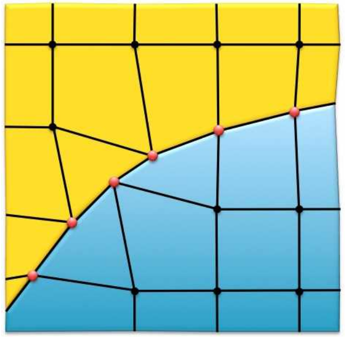
U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

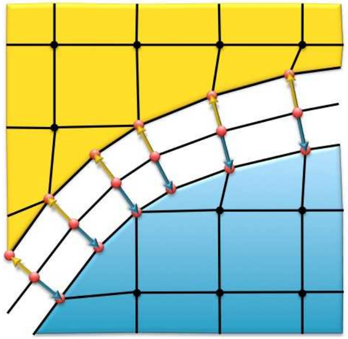
Overlay Grid Hex Meshing Method



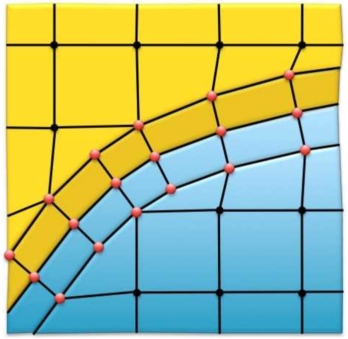
Overlay Cartesian Grid on Geometry



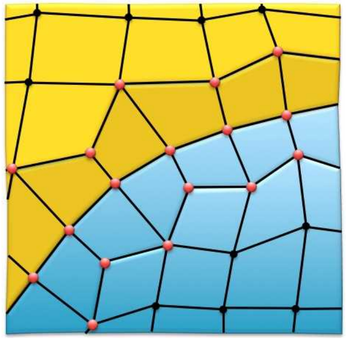
Nodes projected to interfaces



Nodes duplicated at interfaces and moved orthogonally

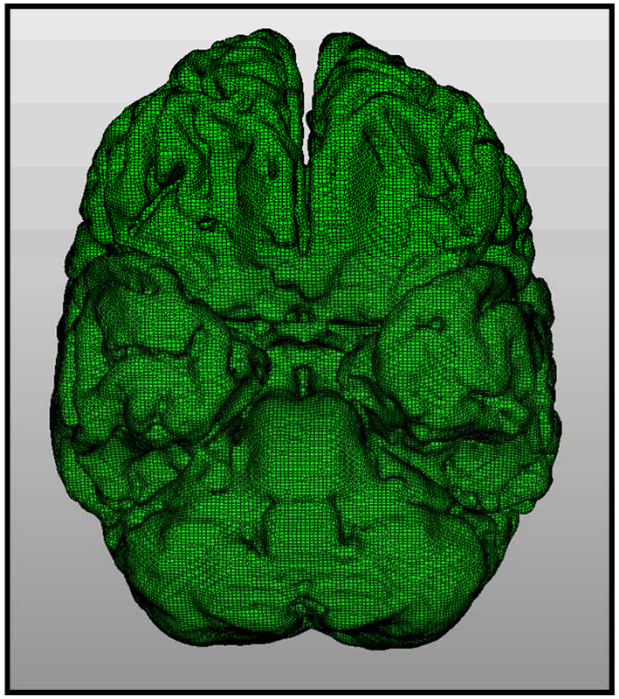


Layers of hexes created at interfaces

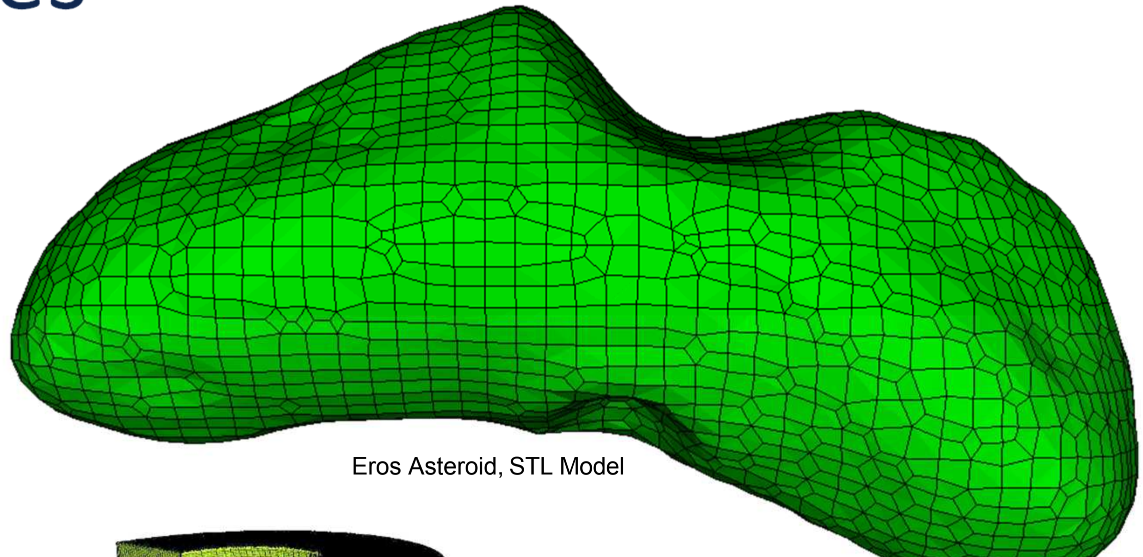


Smoothing performed

Sculpt Examples

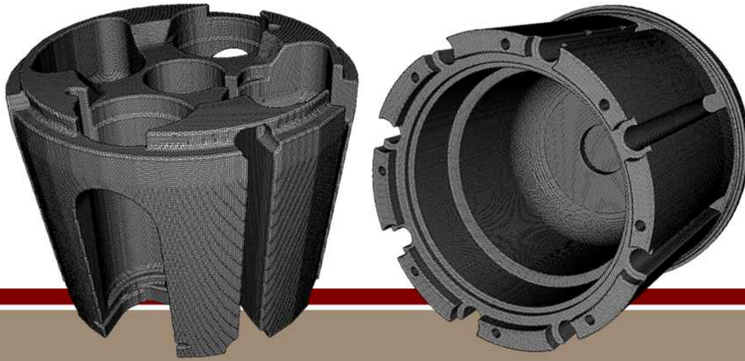


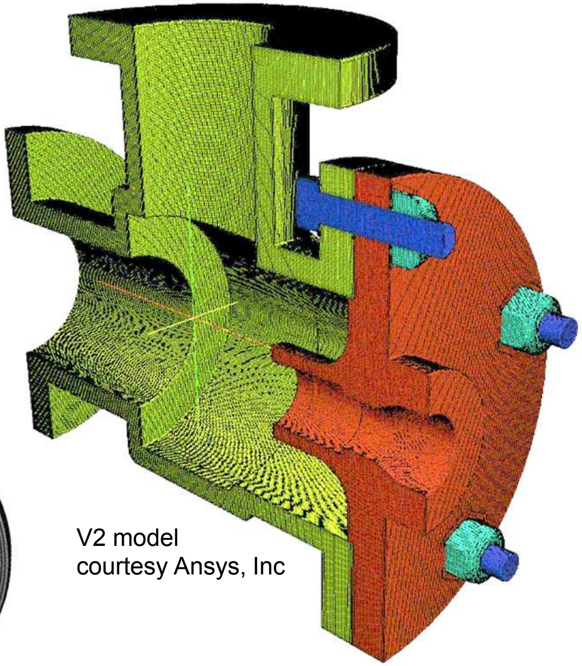
STL MRI Brain Model, Courtesy Bryce Owen
Brigham Young University, Provo, UT



Eros Asteroid, STL Model

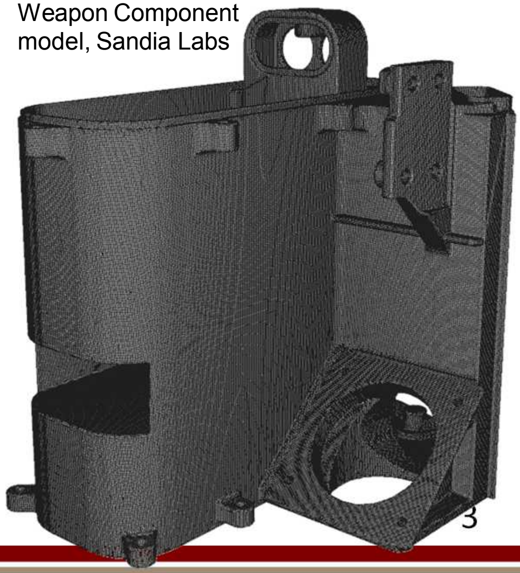
Weapon Component Models Courtesy Stephen Recchia,
US Army, Picitinni. Used with Permission



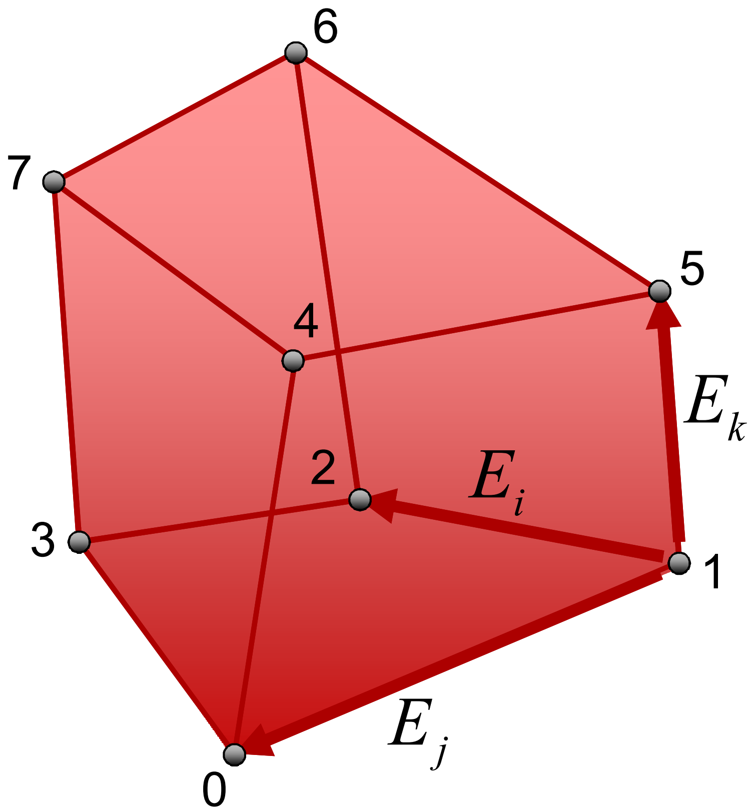


V2 model
courtesy Ansys, Inc

Weapon Component
model, Sandia Labs



Scaled Jacobian



$$(J_s)_I = det\left\{\hat{E}_i\hat{E}_j\hat{E}_k\right\}^{\top}$$

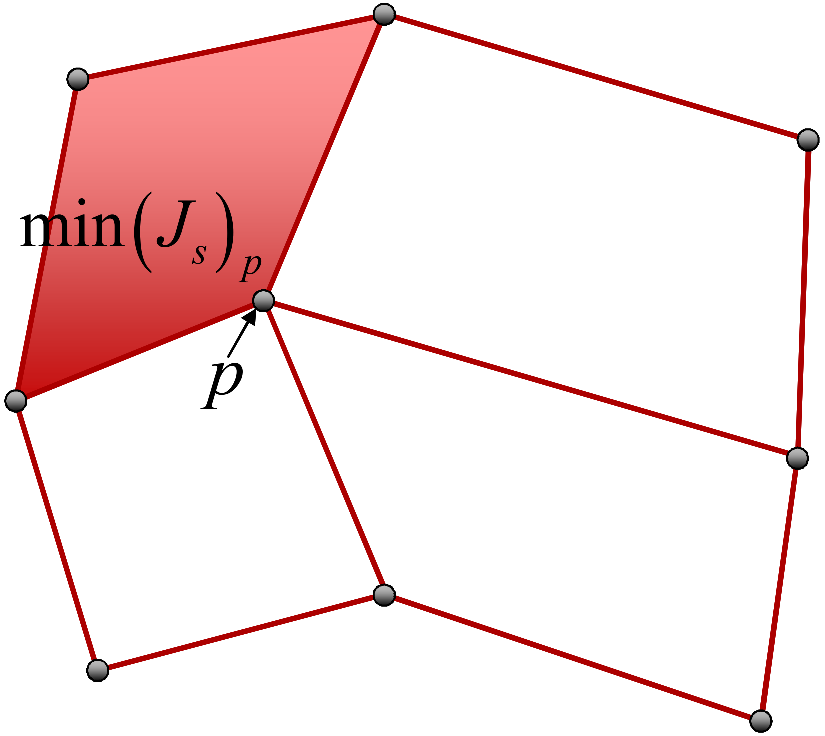
$$J_s = min\left((J_s)_I, I = 0, 1, \ldots 7\right)$$

"Acceptable"

$$J_s \geq 0.2$$

Optimization

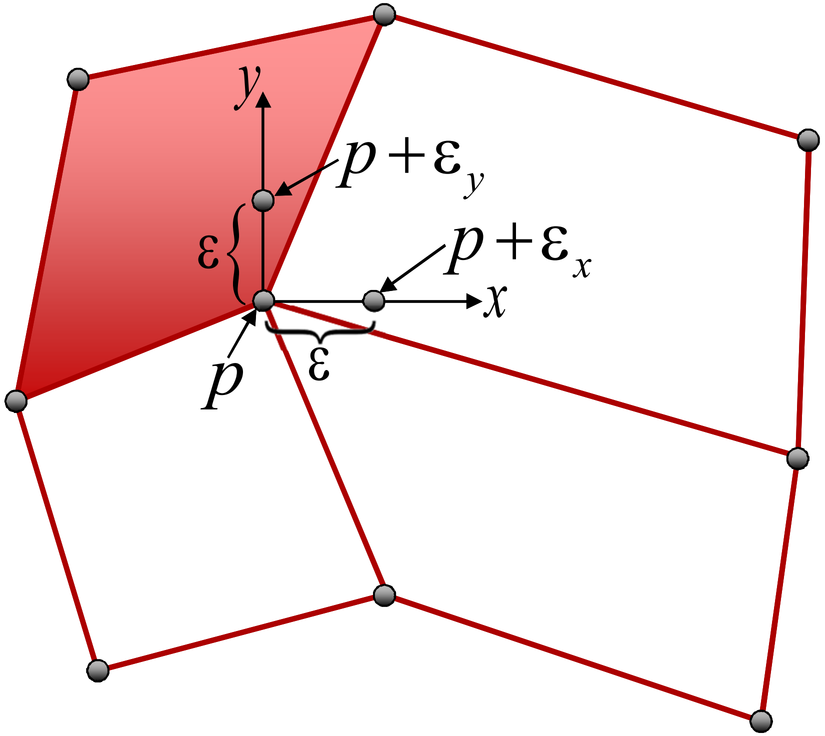
Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes



Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

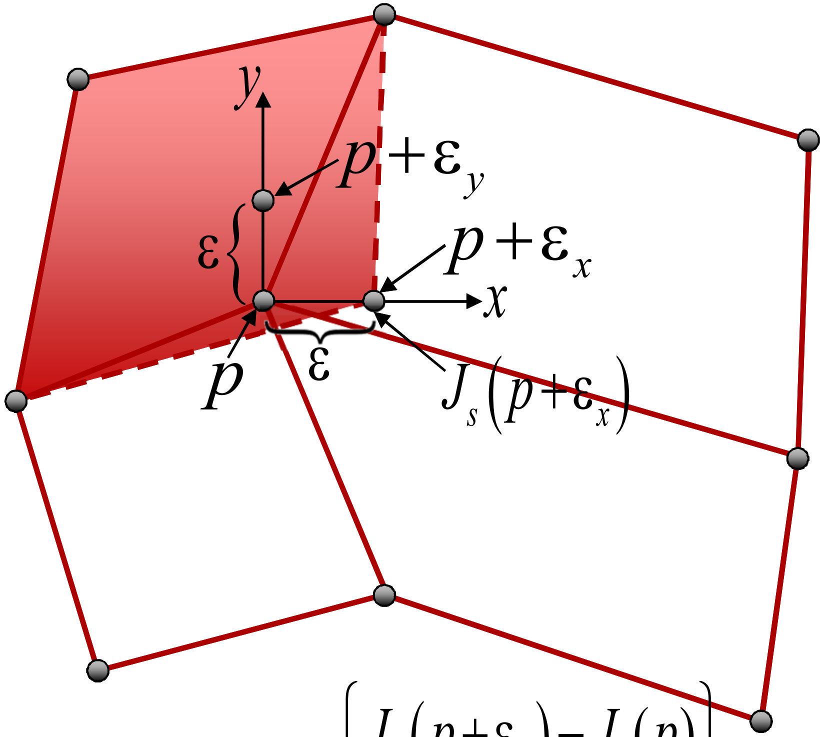
Compute numerical gradient $(\nabla J_s)_p$



Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$

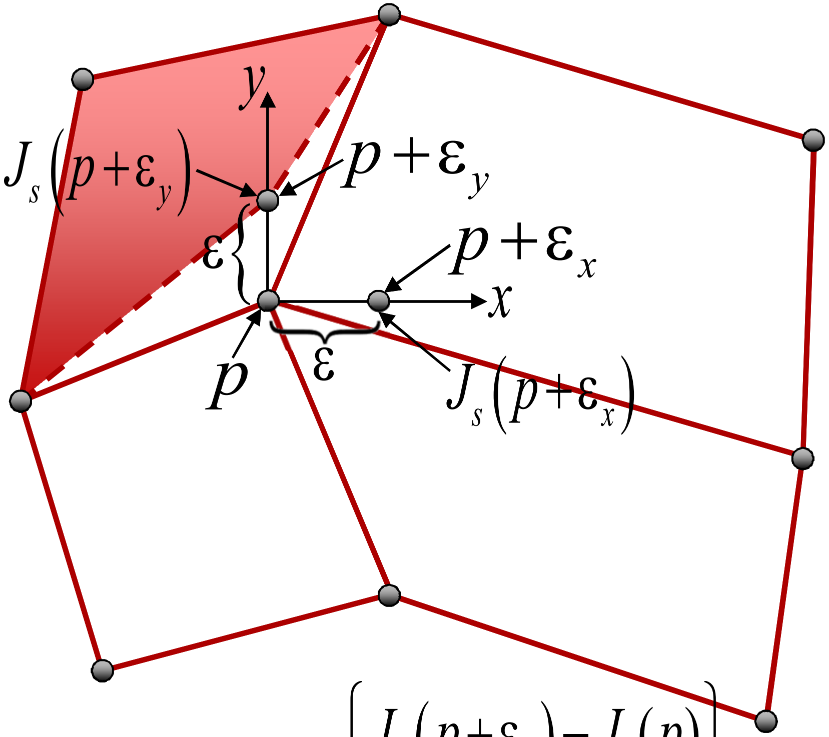


$$(\nabla J_s)_{px} = \left\{ \frac{J_s(p+\varepsilon_x) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$



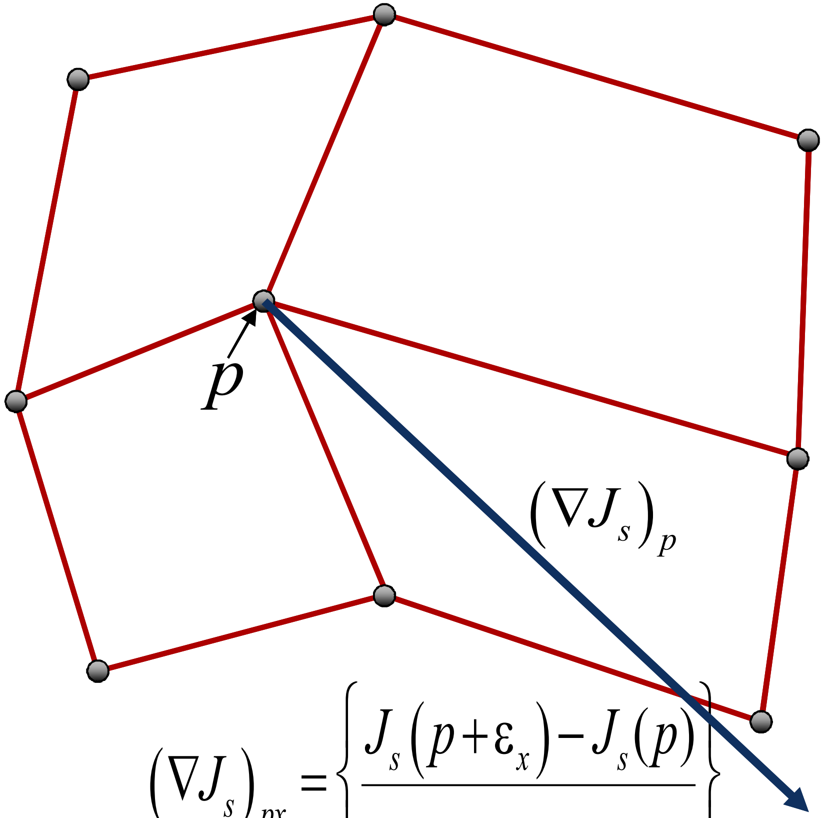
$$(\nabla J_s)_{px} = \left\{\frac{J_s(p+\varepsilon_x)-J_s(p)}{\varepsilon}\right\}$$

$$(\nabla J_s)_{py} = \left\{\frac{J_s(p+\varepsilon_y)-J_s(p)}{\varepsilon}\right\}$$

Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$



$$(\nabla J_s)_{px} = \left\{ \frac{J_s(p+\varepsilon_x) - J_s(p)}{\varepsilon} \right\}$$

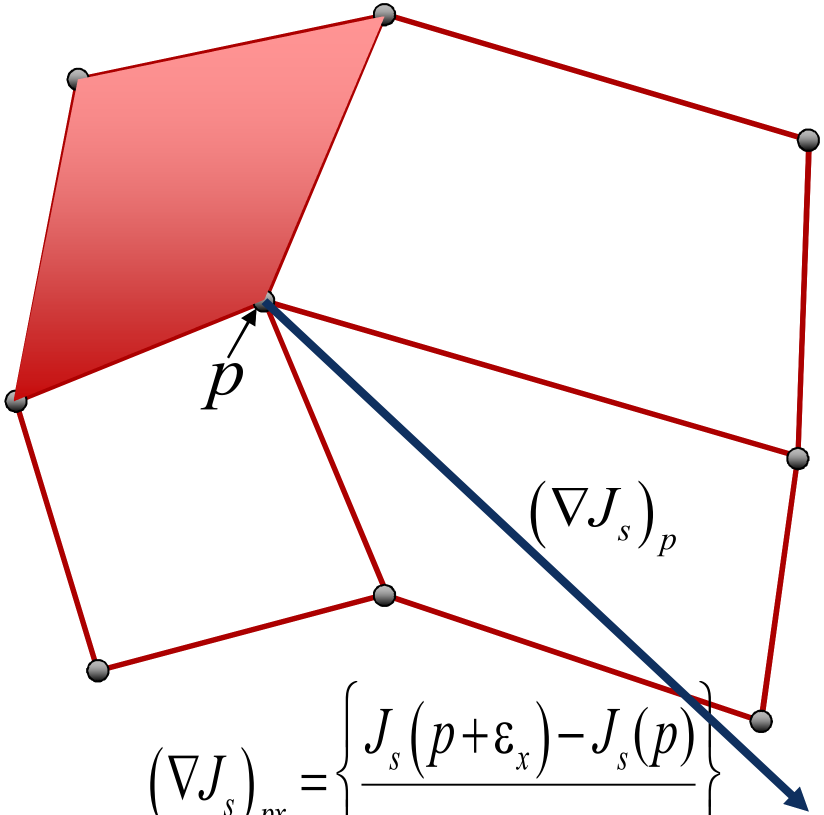
$$(\nabla J_s)_{py} = \left\{ \frac{J_s(p+\varepsilon_y) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $\left(J_s\right)_p$, of node p in all attached hexes

Compute numerical gradient $\left(\nabla J_s\right)_p$

Find improved $\left(J_s\right)_p$ by searching along vector $\left(\nabla J_s\right)_p$



$$\left(\nabla J_s\right)_{px} = \left\{ \frac{J_s\left(p+\varepsilon_x\right) - J_s(p)}{\varepsilon} \right\}$$

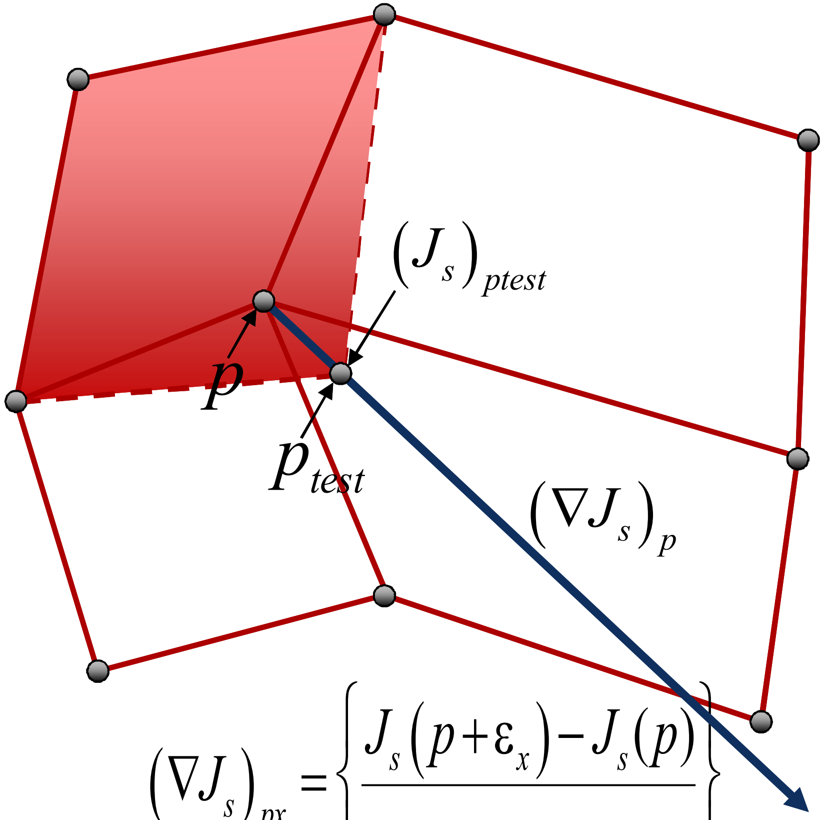
$$\left(\nabla J_s\right)_{py} = \left\{ \frac{J_s\left(p+\varepsilon_y\right) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$

Find improved $(J_s)_p$ by searching along vector $(\nabla J_s)_p$



$$(\nabla J_s)_{px} = \left\{ \frac{J_s(p+\varepsilon_x) - J_s(p)}{\varepsilon} \right\}$$

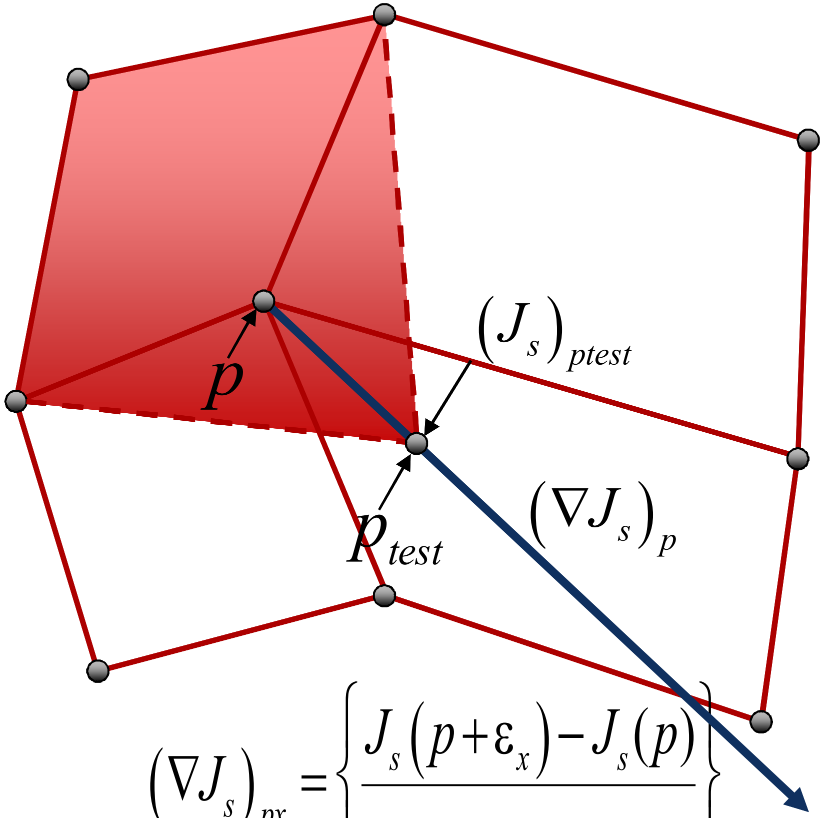
$$(\nabla J_s)_{py} = \left\{ \frac{J_s(p+\varepsilon_y) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$

Find improved $(J_s)_p$ by searching along vector $(\nabla J_s)_p$



$$(\nabla J_s)_{px} = \left\{ \frac{J_s(p+\varepsilon_x) - J_s(p)}{\varepsilon} \right\}$$

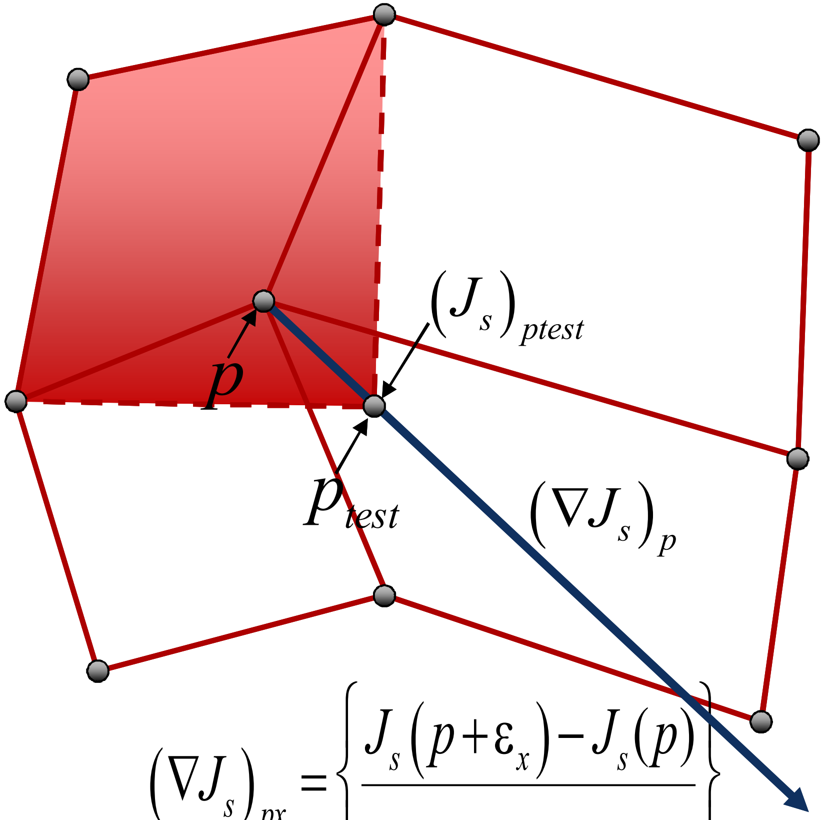
$$(\nabla J_s)_{py} = \left\{ \frac{J_s(p+\varepsilon_y) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $(J_s)_p$, of node p in all attached hexes

Compute numerical gradient $(\nabla J_s)_p$

Find improved $(J_s)_p$ by searching along vector $(\nabla J_s)_p$



$$(\nabla J_s)_{px} = \left\{ \frac{J_s(p+\varepsilon_x) - J_s(p)}{\varepsilon} \right\}$$

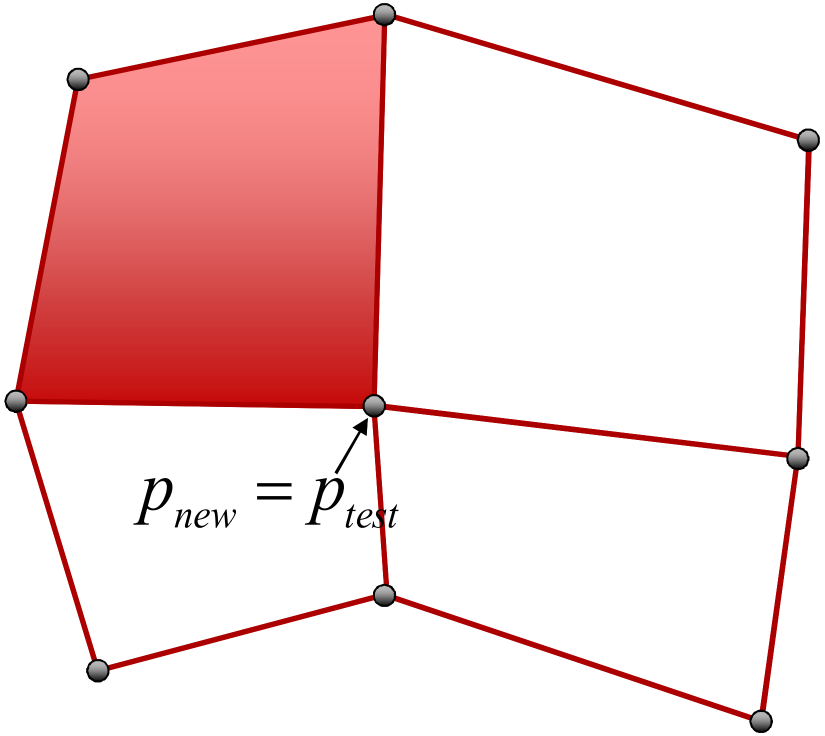
$$(\nabla J_s)_{py} = \left\{ \frac{J_s(p+\varepsilon_y) - J_s(p)}{\varepsilon} \right\}$$

Optimization

Compute minimum scaled Jacobian, $\left(J_s\right)_p$, of node p in all attached hexes

Compute numerical gradient $\left(\nabla J_s\right)_p$

Find improved $\left(J_s\right)_p$ by searching along vector $\left(\nabla J_s\right)_p$



Optimization

Compute minimum scaled Jacobian, $\left(J_s\right)_p$, of node p in all attached hexes

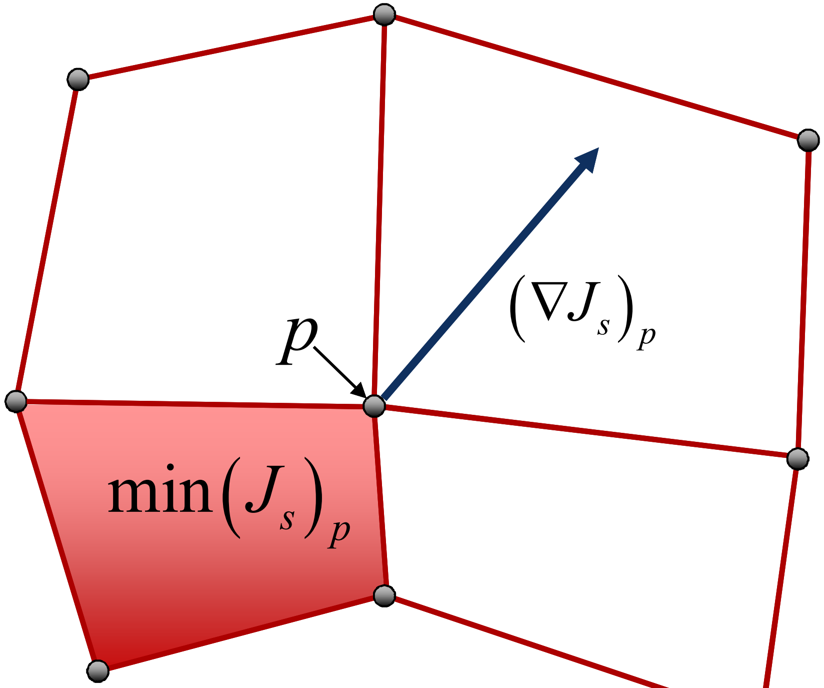
Compute numerical gradient $\left(\nabla J_s\right)_p$

Find improved $\left(J_s\right)_p$ by searching along vector $\left(\nabla J_s\right)_p$

Stopping Criteria

$$\left(J_s\right)_p > 0.2$$

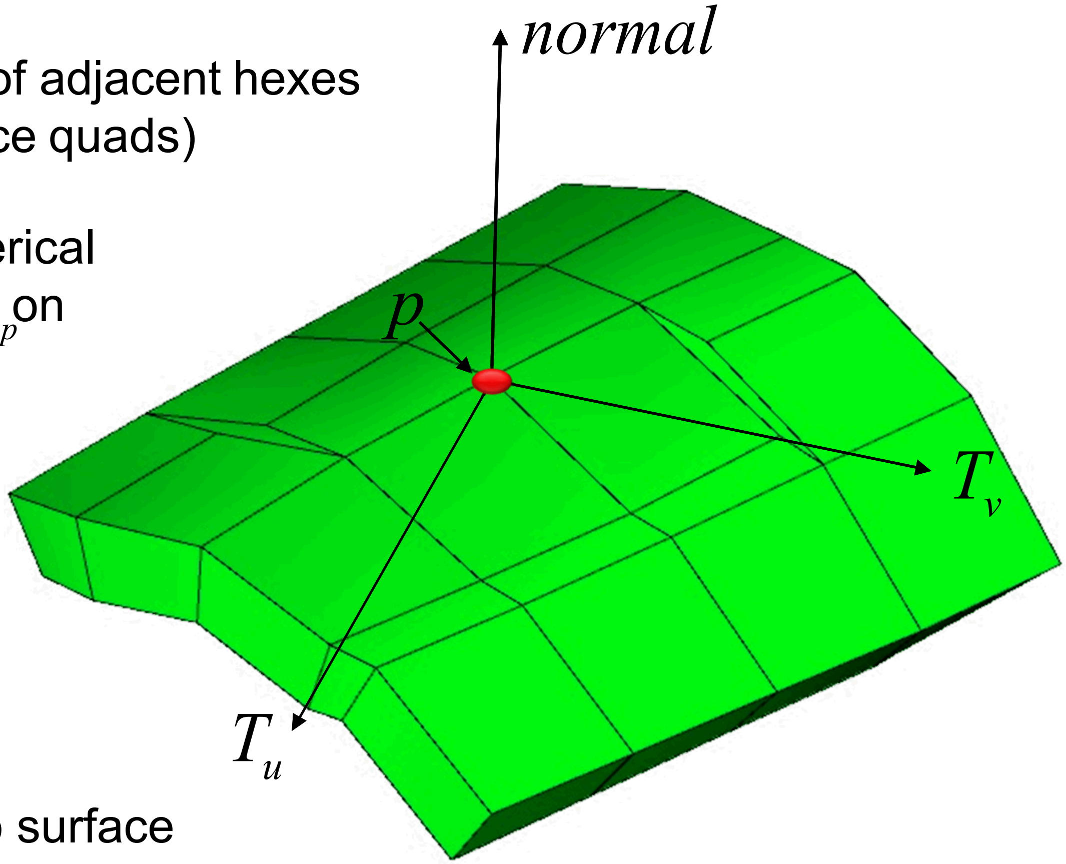
Maximum 3 iterations



Optimization for Surface Nodes

Optimize $\left(J_s\right)_p$ of adjacent hexes to p (not surface quads)

Compute numerical gradient $\left(\nabla J_s\right)_p$ on tangent plane

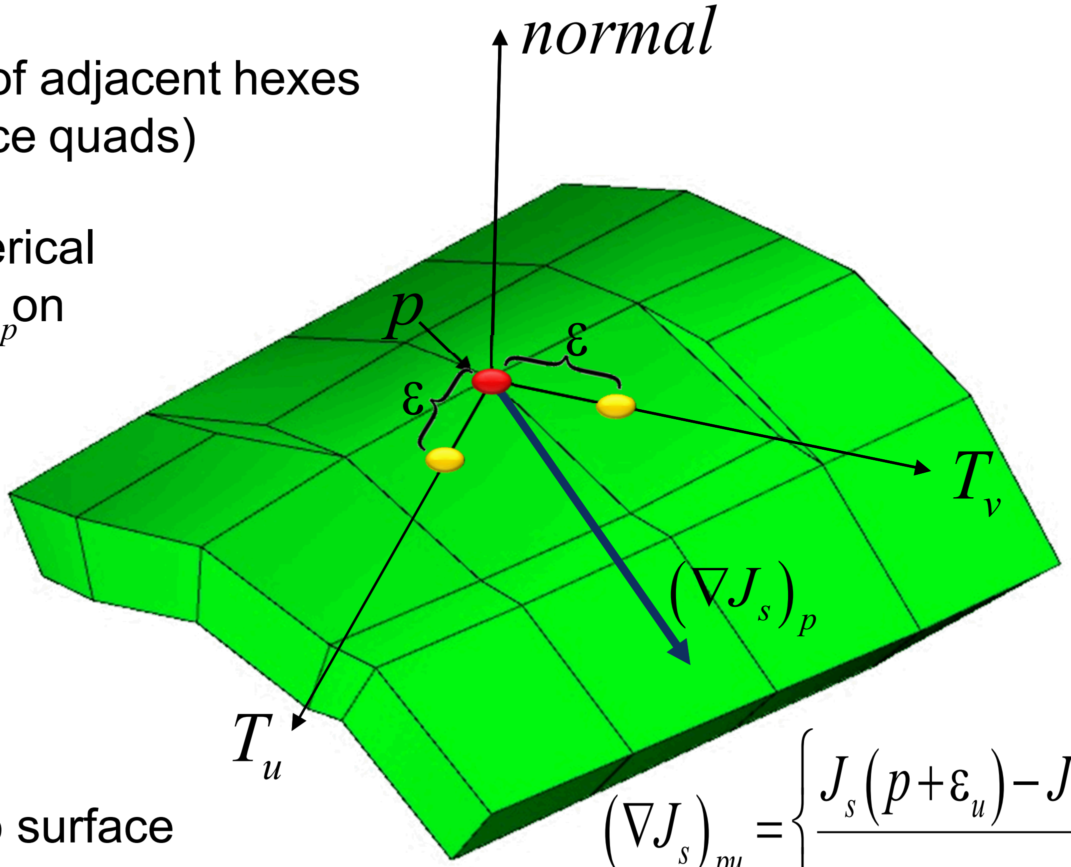


Project p_{new} to surface

Optimization for Surface Nodes

Optimize $\left(J_s\right)_p$ of adjacent hexes to p (not surface quads)

Compute numerical gradient $\left(\nabla J_s\right)_p$ on tangent plane

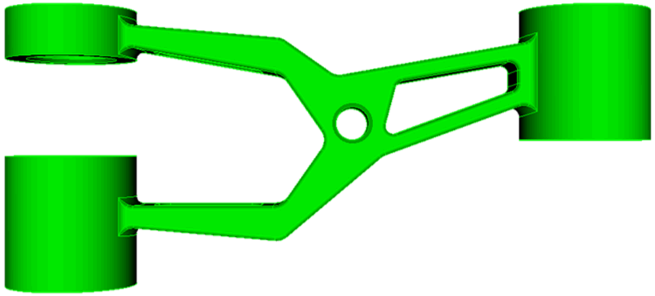


Project p_{new} to surface

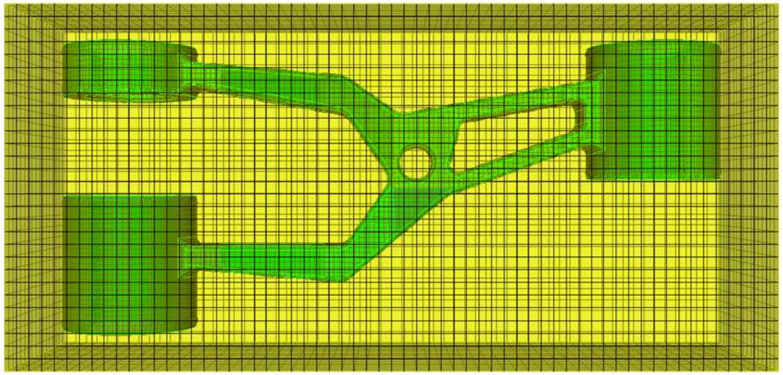
$$\left(\nabla J_s\right)_{pu} = \left\{ \frac{J_s\left(p+\varepsilon_u\right) - J_s(p)}{\varepsilon} \right\}$$

$$\left(\nabla J_s\right)_{pv} = \left\{ \frac{J_s\left(p+\varepsilon_v\right) - J_s(p)}{\varepsilon} \right\}$$

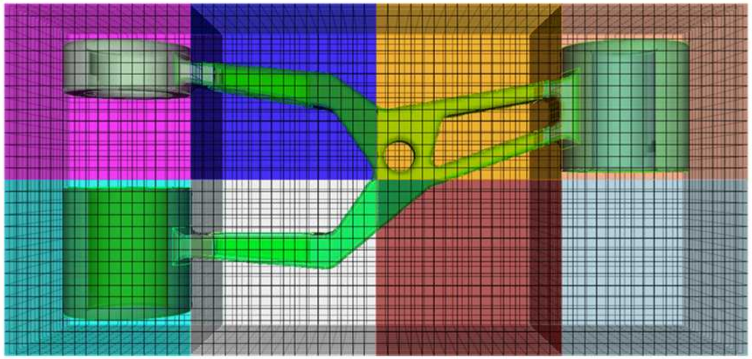
Distributed Meshing



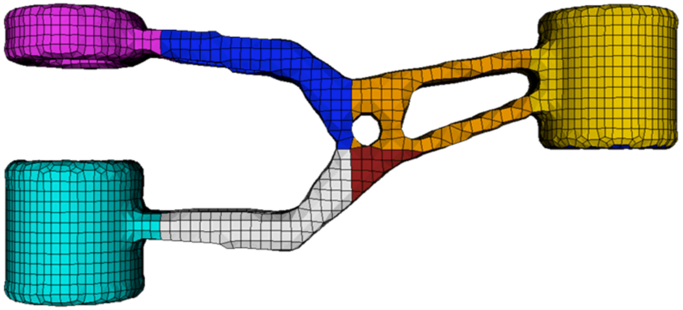
CAD geometry



Global overlay Cartesian grid

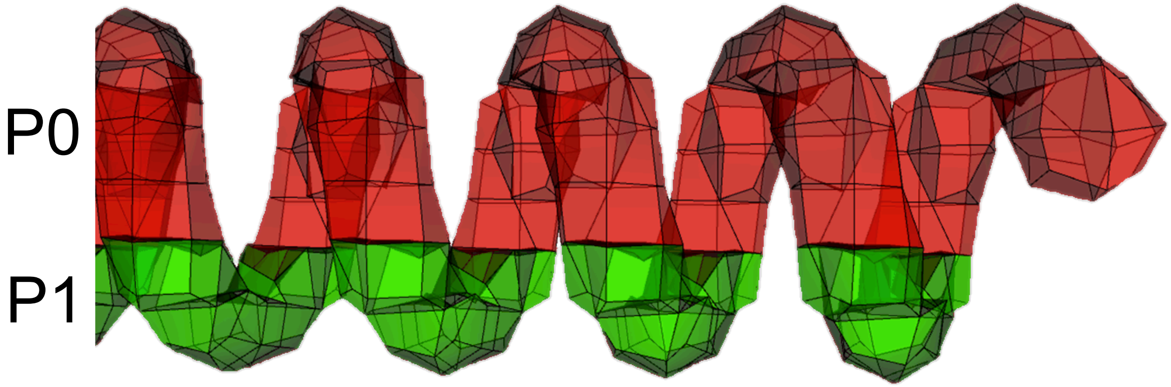


Cartesian grid decomposed and distributed amongst many processors

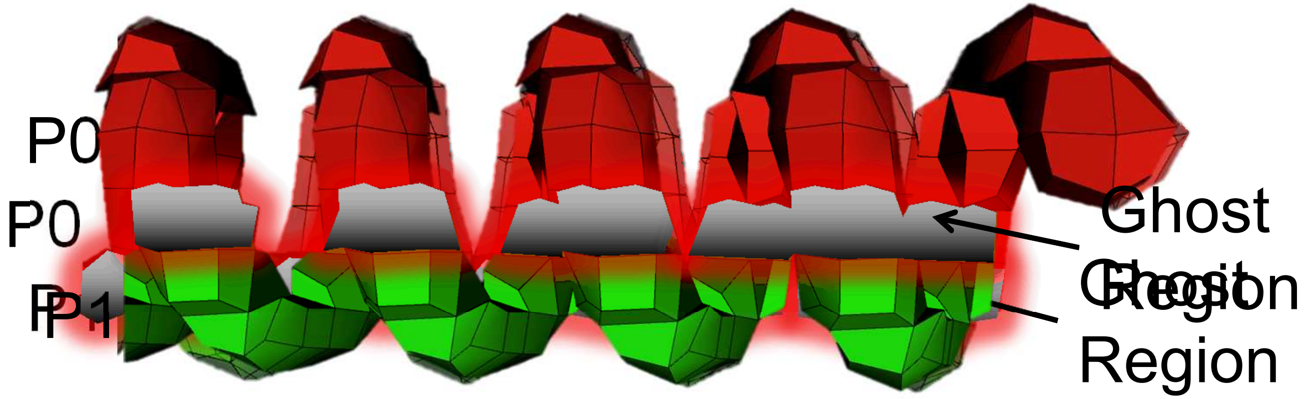


Each processor independently meshes its portion of Cartesian grid

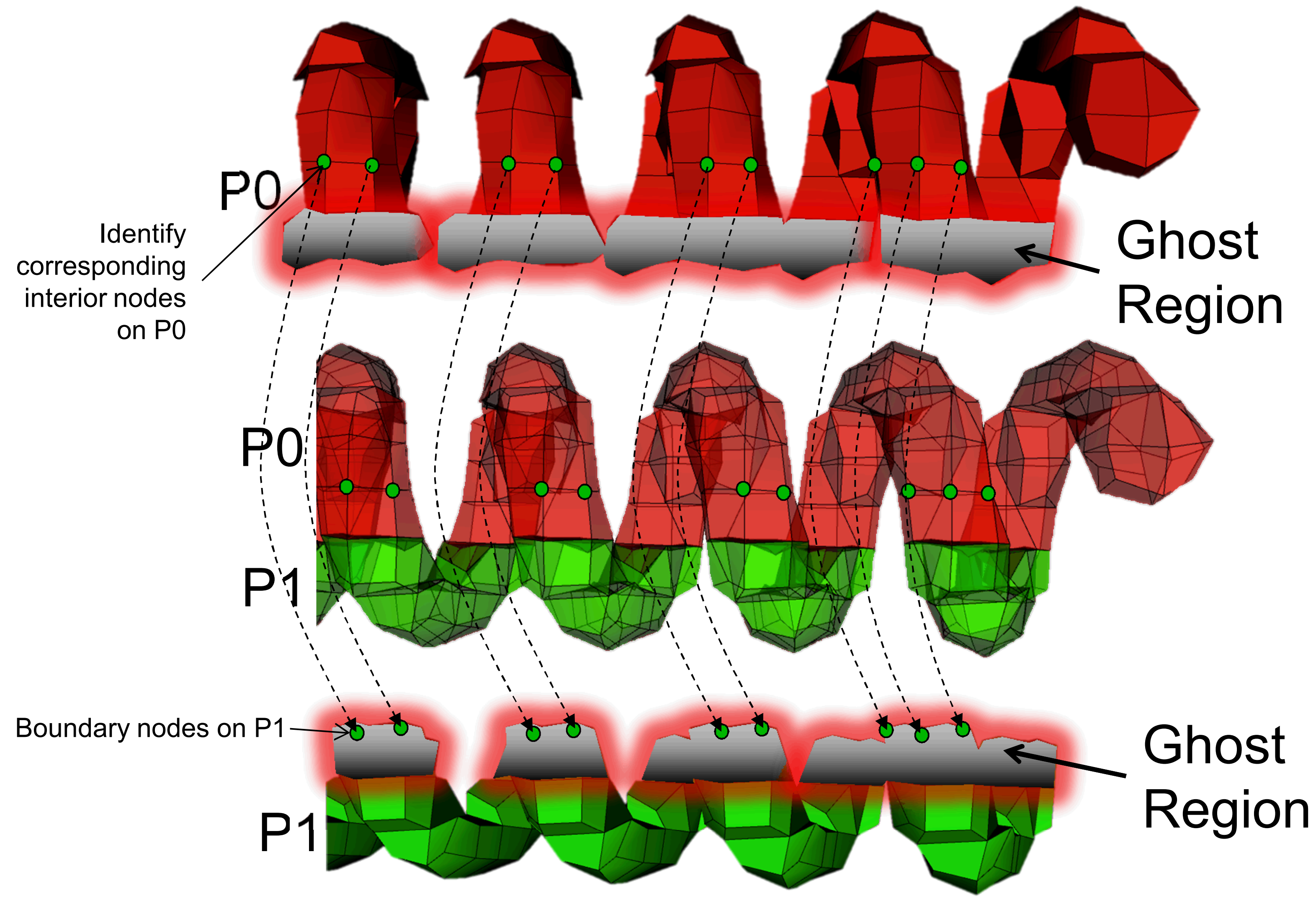
Communication for Smoothing



Communication for Smoothing



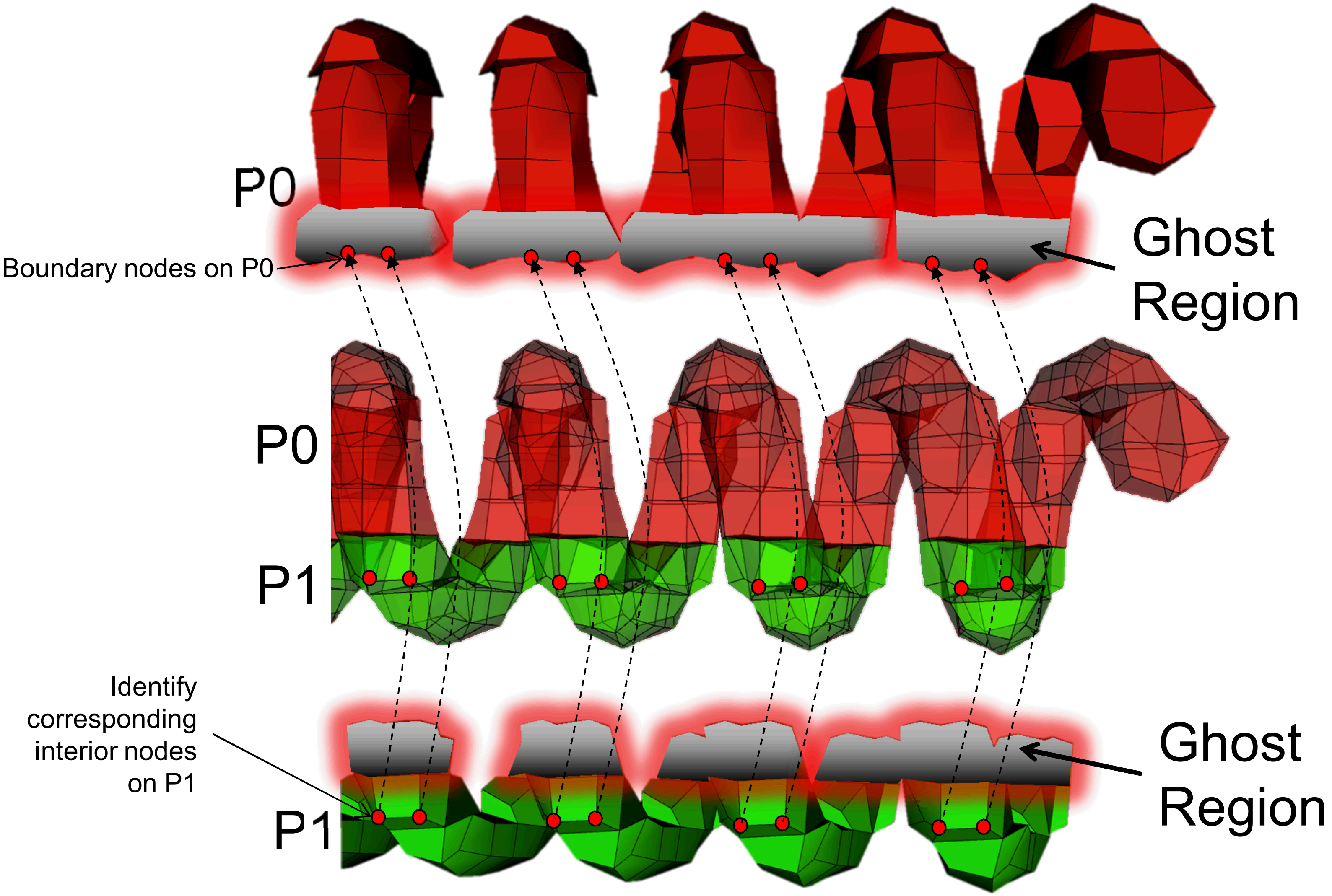
Communication for Smoothing



$L^{0\to 1}$

Node locations on P0 used to update boundary nodes on P1

Communication for Smoothing



$L^{0\to1}$

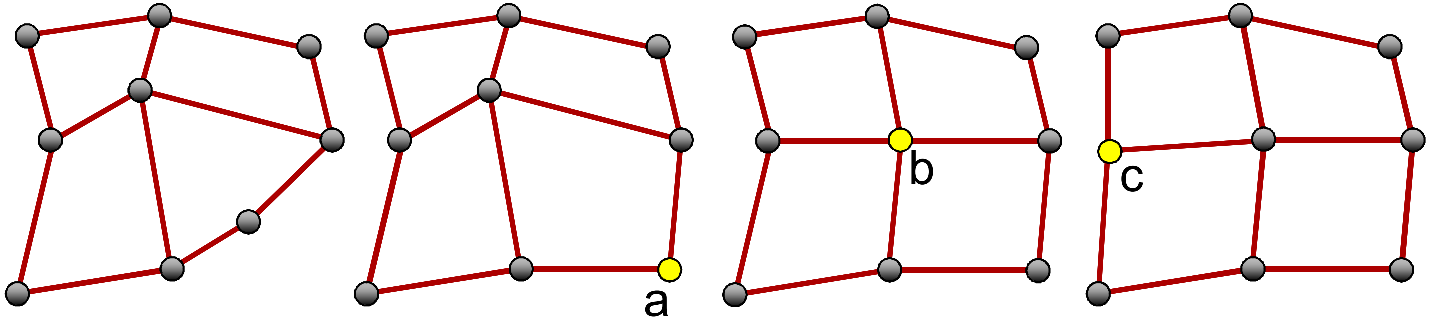
Node locations on P0 used to update boundary nodes on P1

$L^{1\to0}$

Node locations on P1 used to update boundary nodes on P0

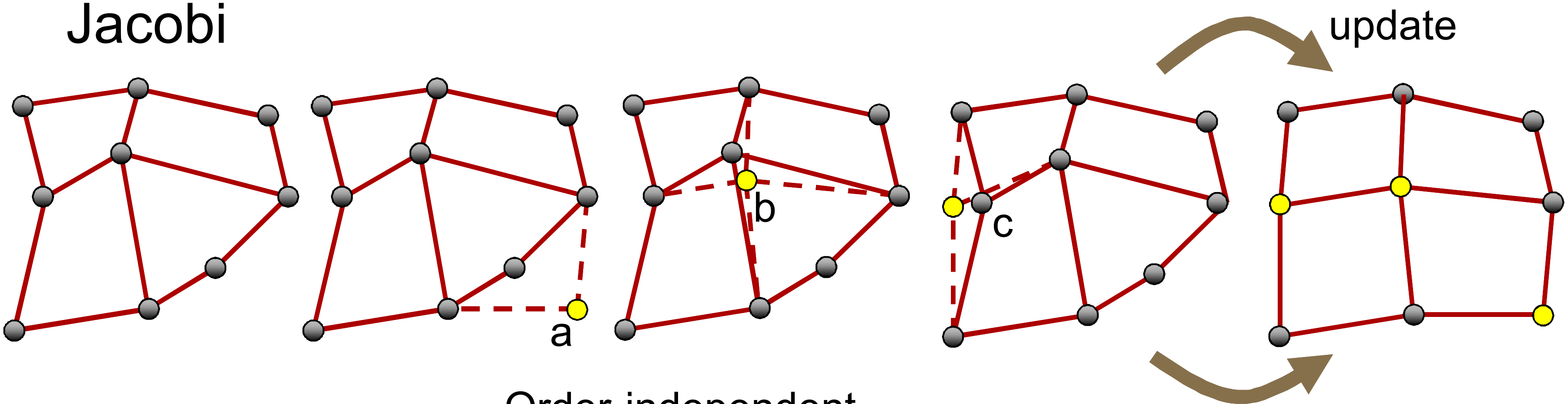
Smoothing Strategies

Gauss-Siedel



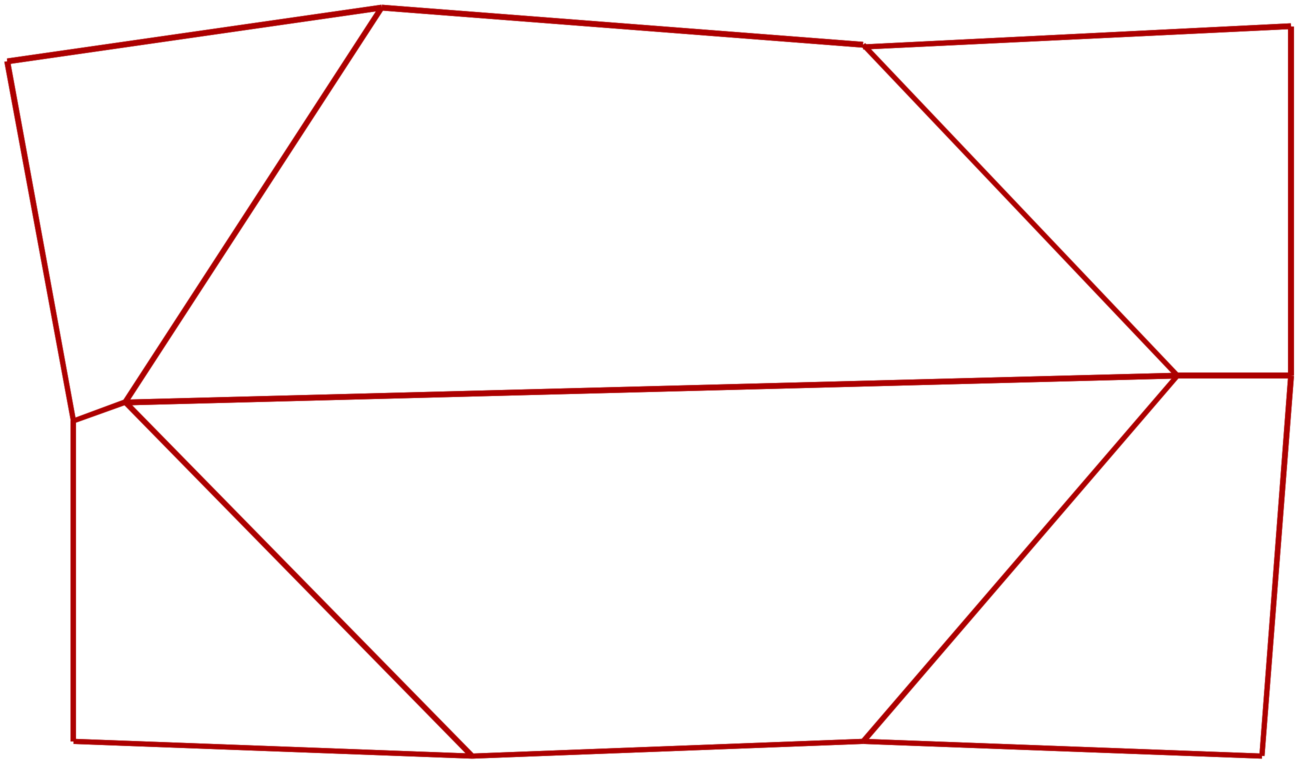
Order-dependent

Jacobi

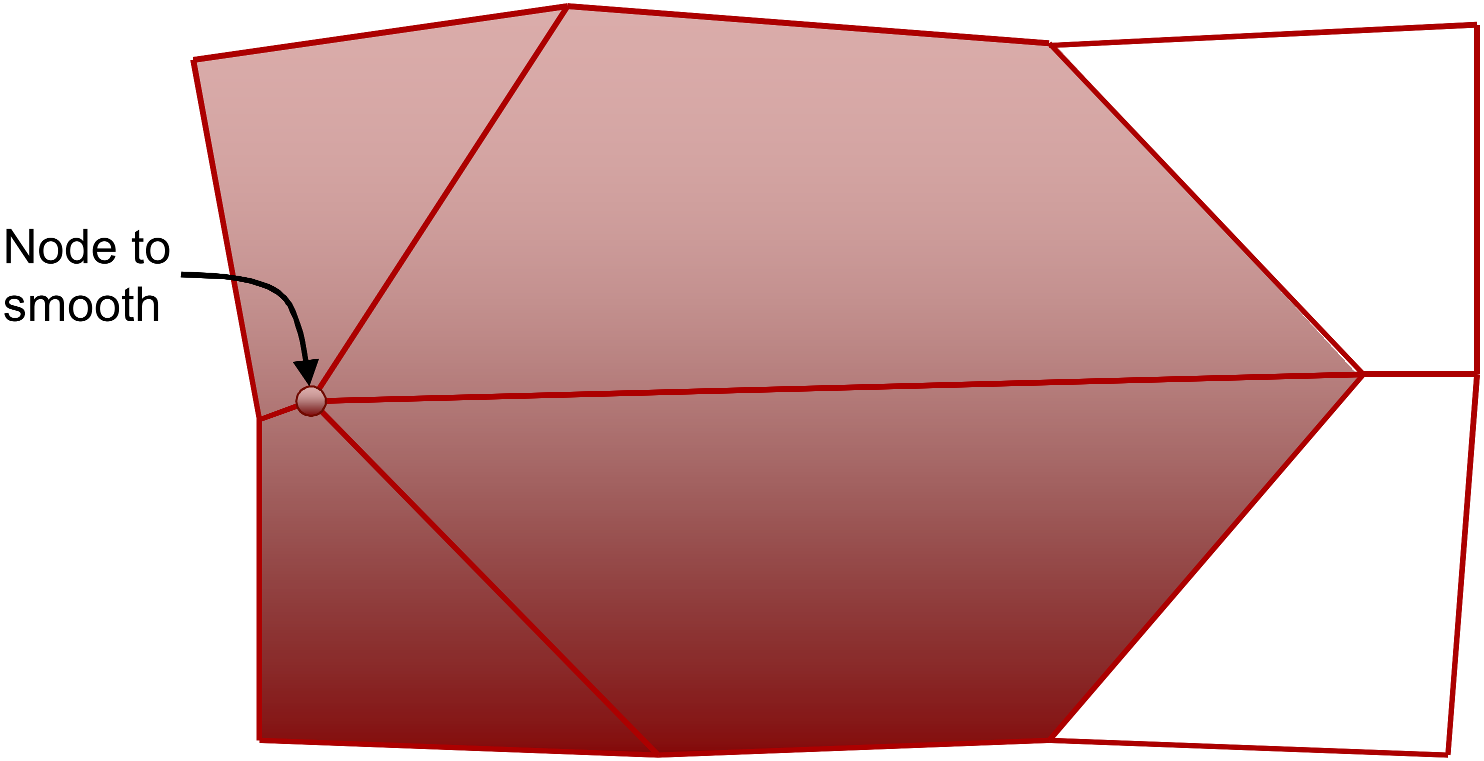


Order-independent

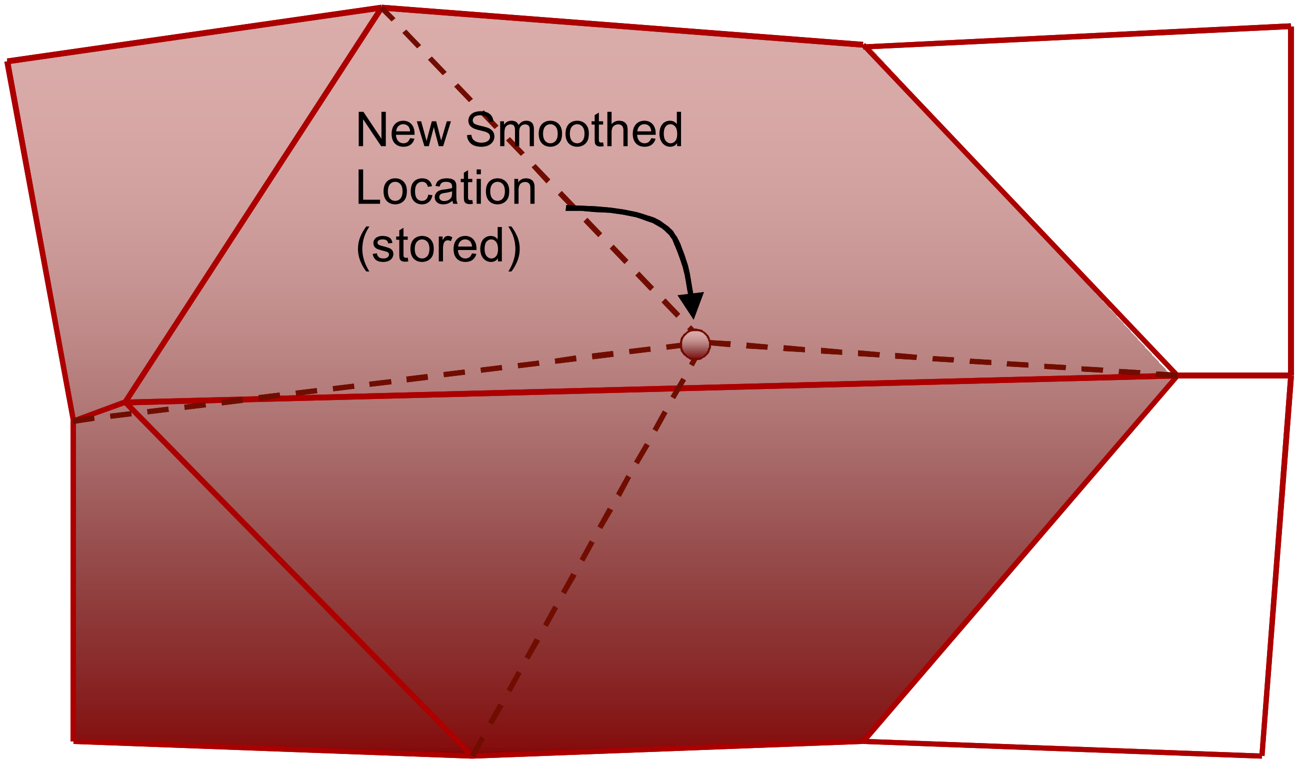
Jacobi Smoothing



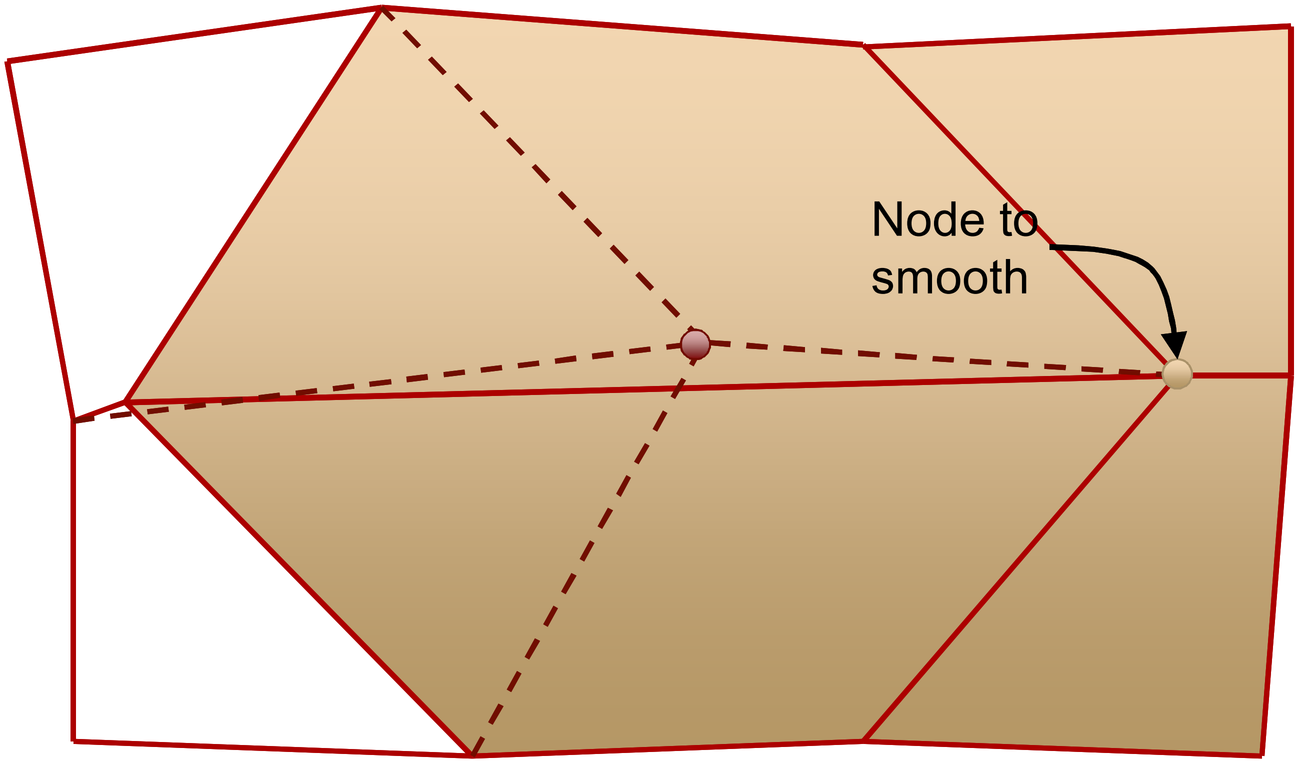
Jacobi Smoothing



Jacobi Smoothing

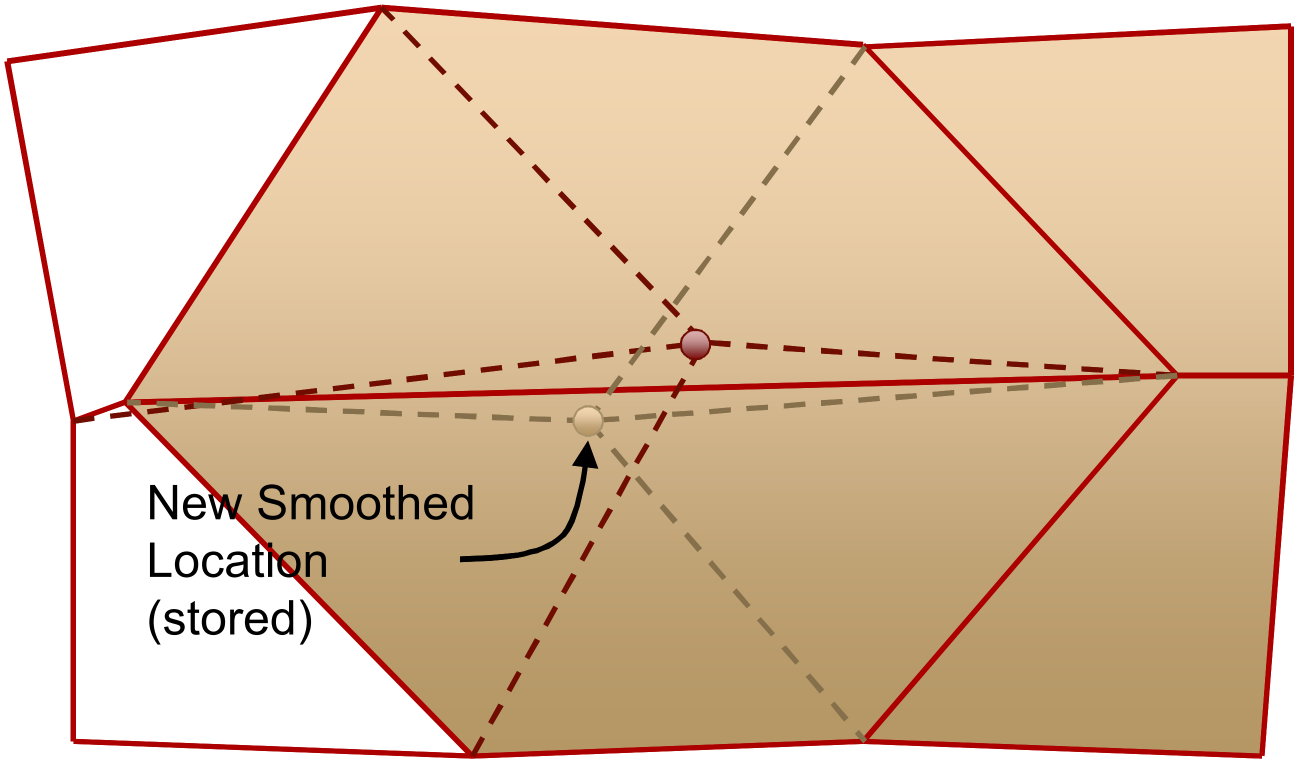


Jacobi Smoothing



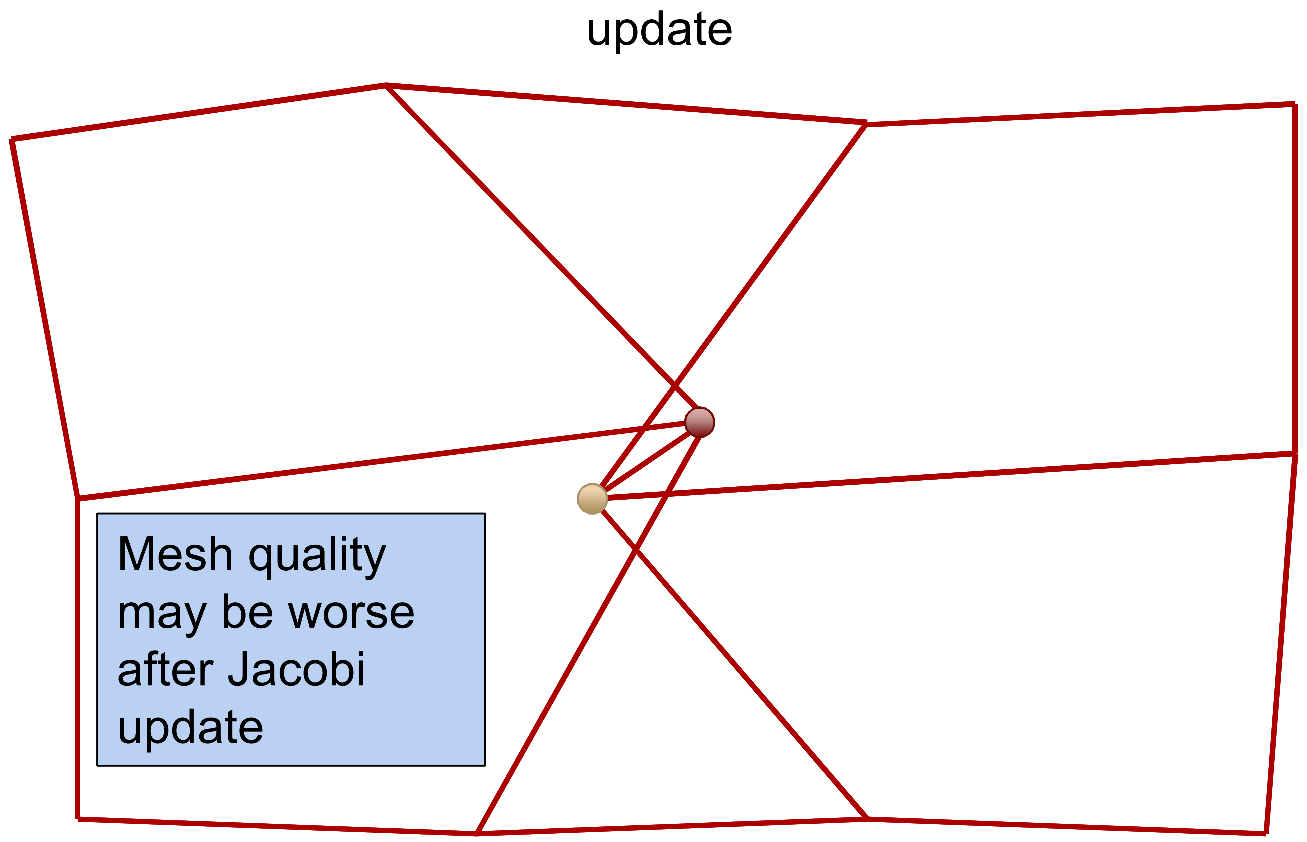
Node to smooth

Jacobi Smoothing



New Smoothed Location (stored)

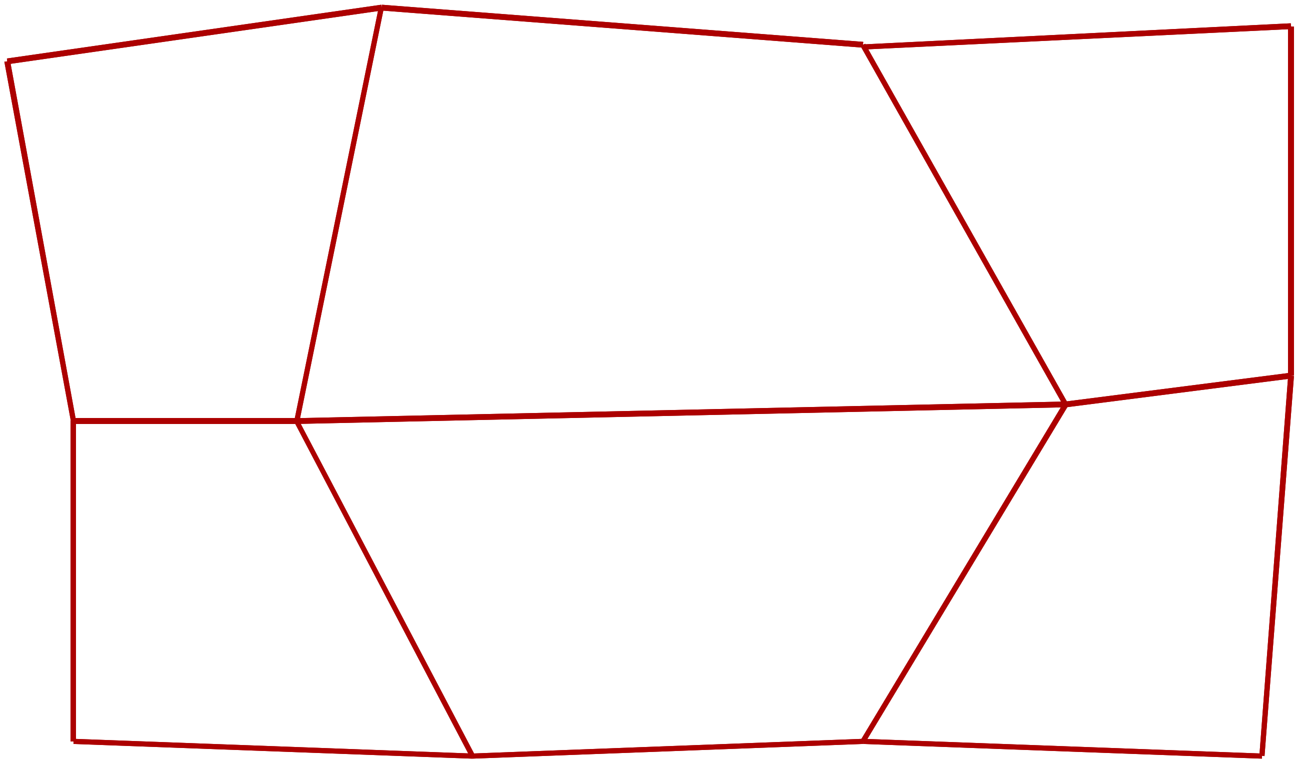
Jacobi Smoothing



update

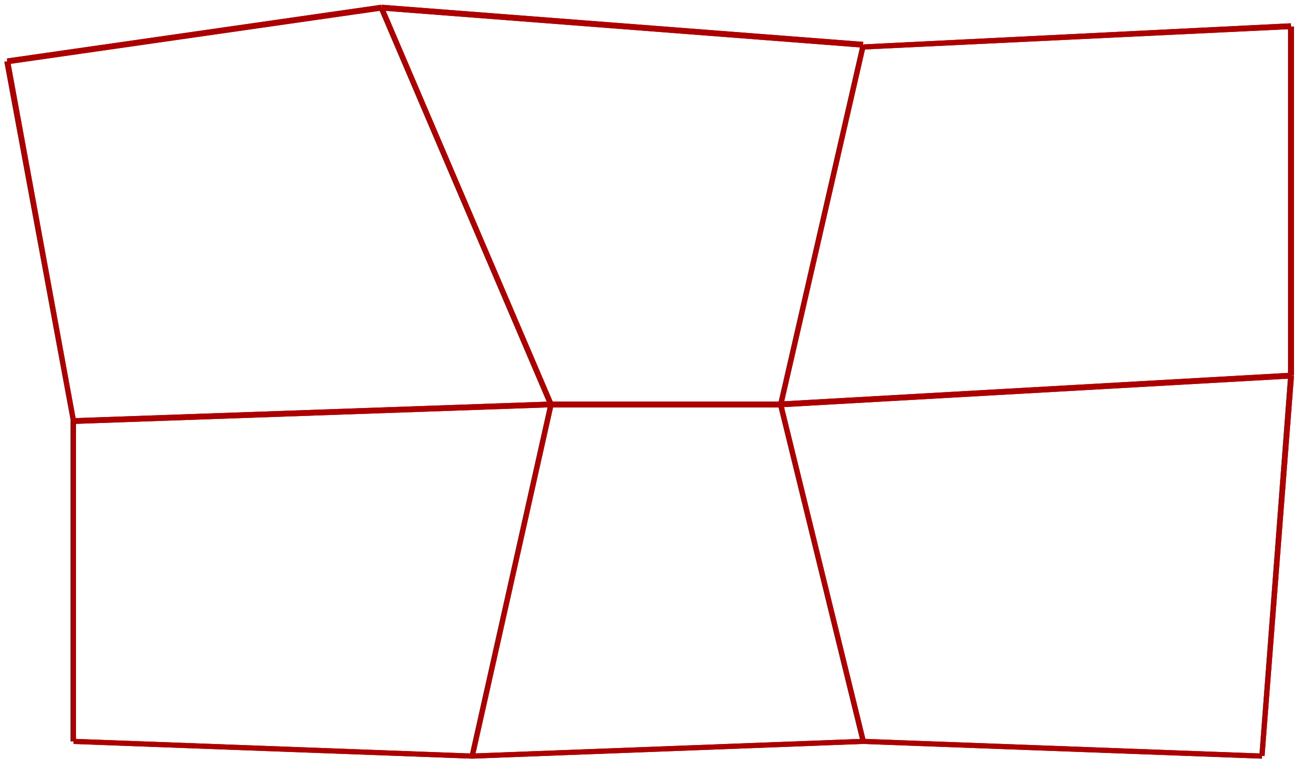
Mesh quality may be worse after Jacobi update

Jacobi Smoothing



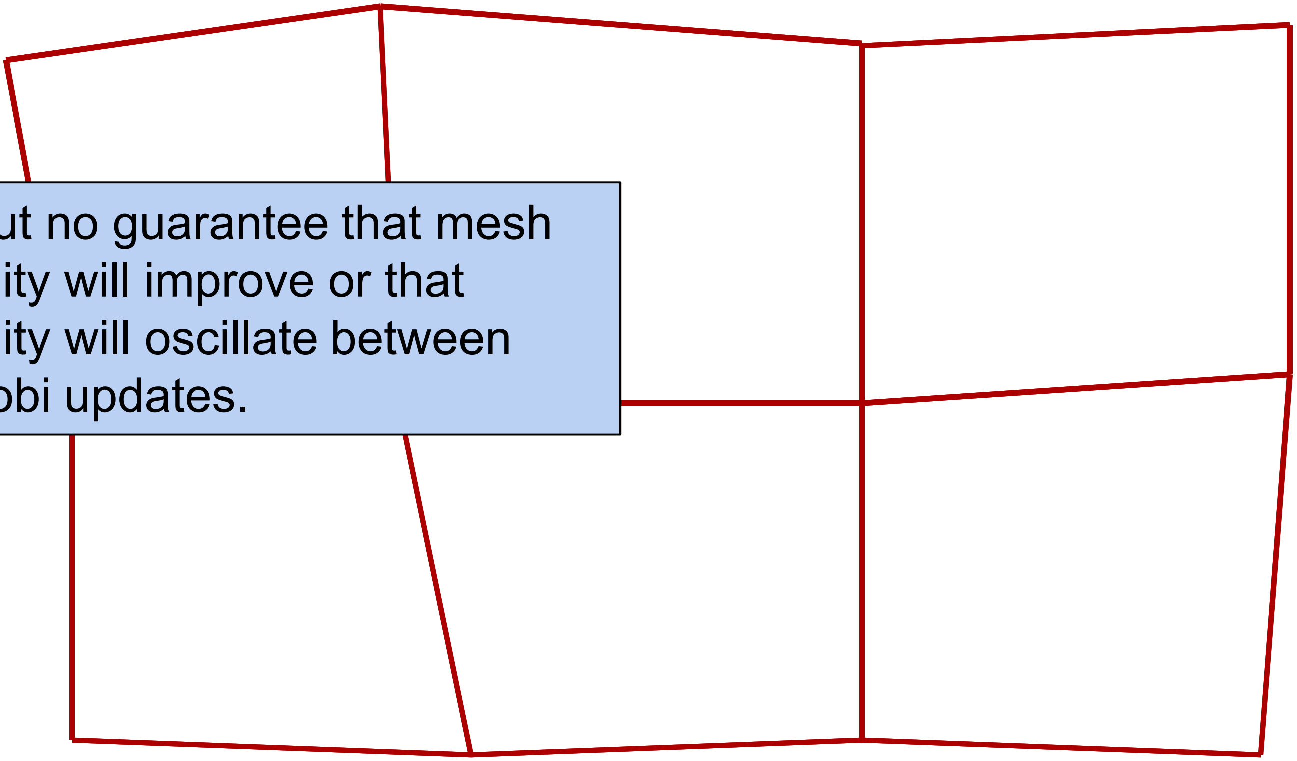
After several iterations, quality in *most* cases will improve

Jacobi Smoothing



After several iterations, quality in *most* cases will improve

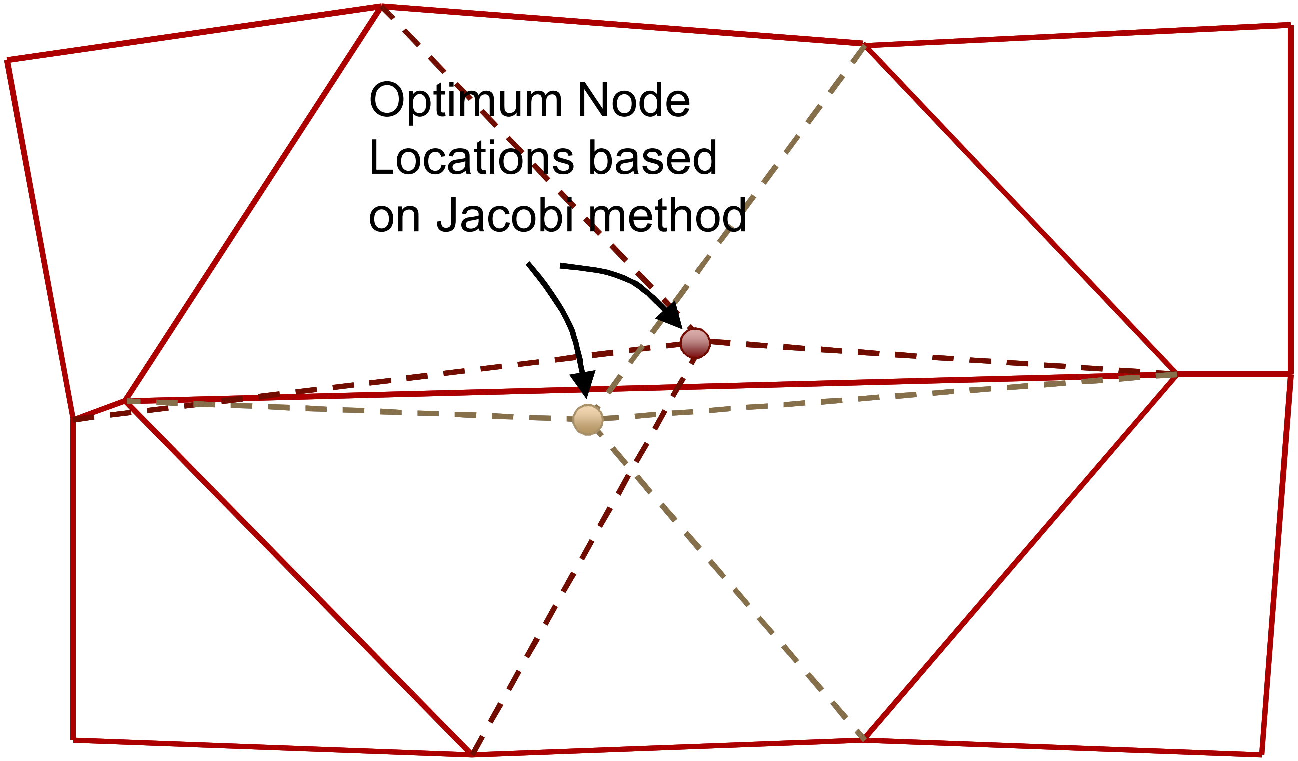
Jacobi Smoothing



…But no guarantee that mesh quality will improve or that quality will oscillate between Jacobi updates.

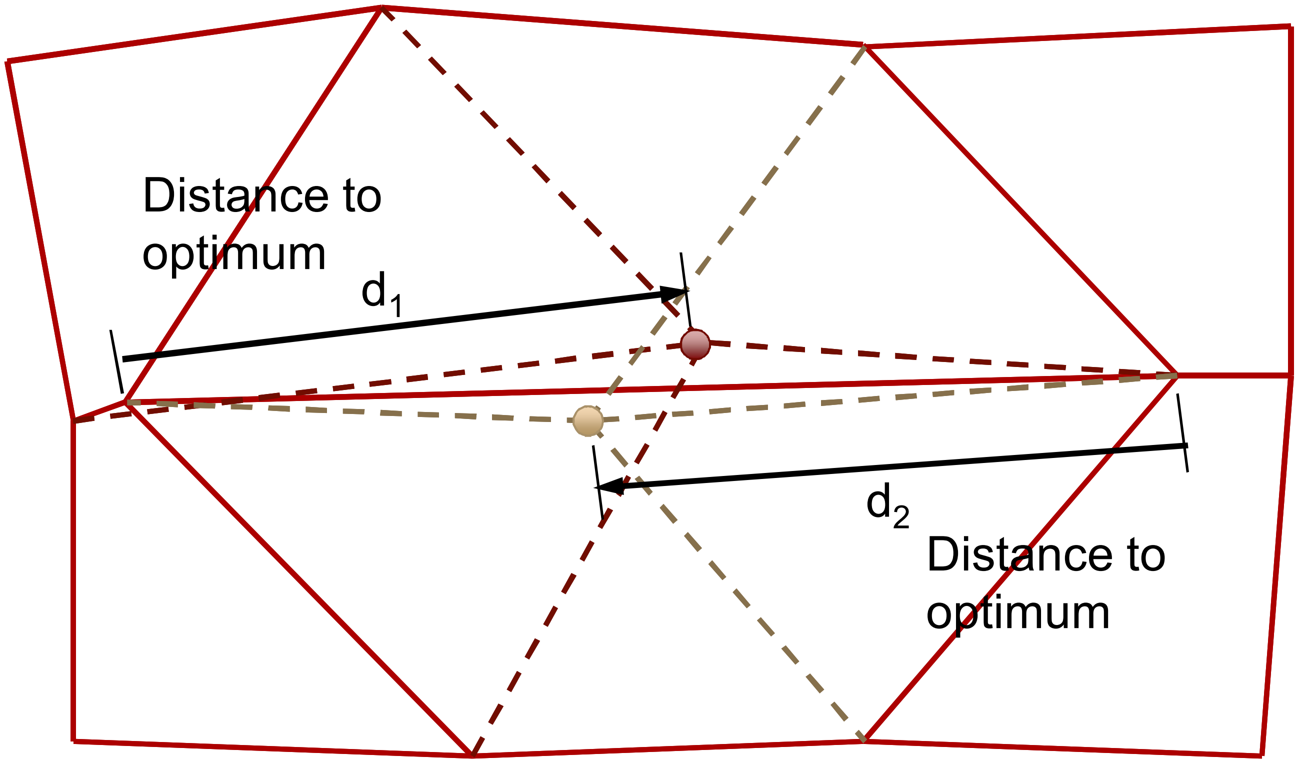
After several iterations, quality in *most* cases will improve

Jacobi Smoothing



Solution: Apply *damping* to node movement

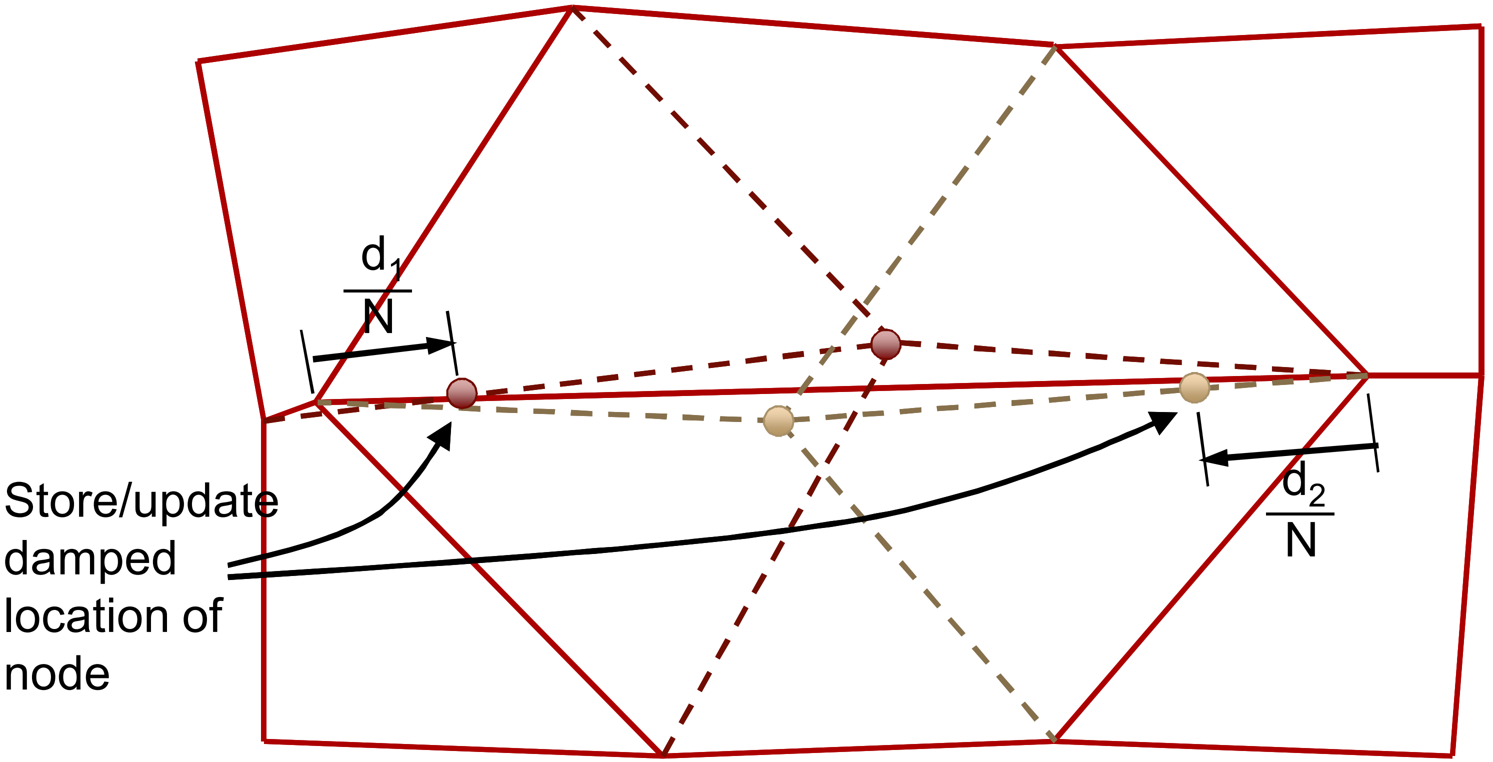
Jacobi Smoothing



Solution: Apply *damping* to node movement

Jacobi Smoothing

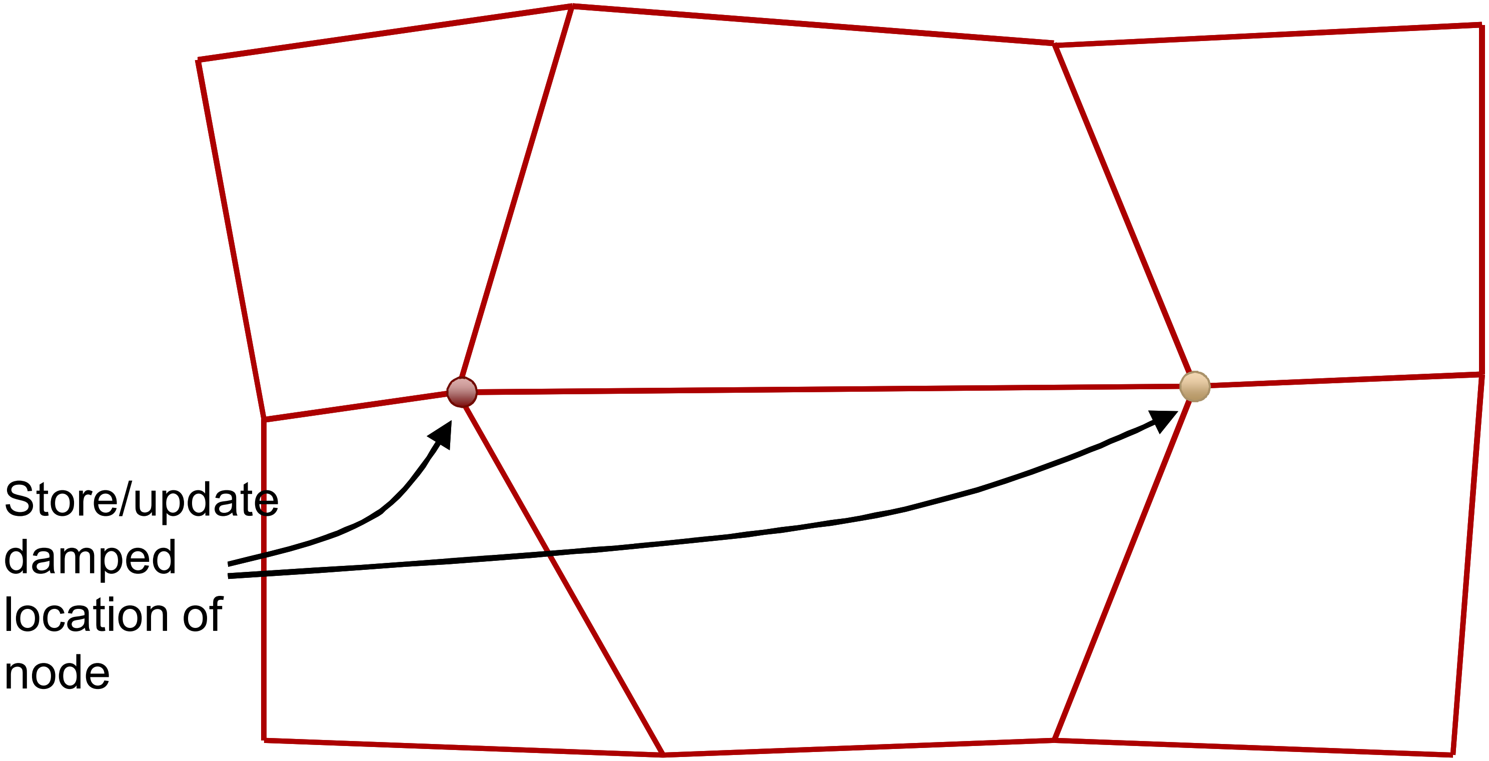
N = number of remaining smoothing iterations



Store/update damped location of node

Solution: Apply *damping* to node movement

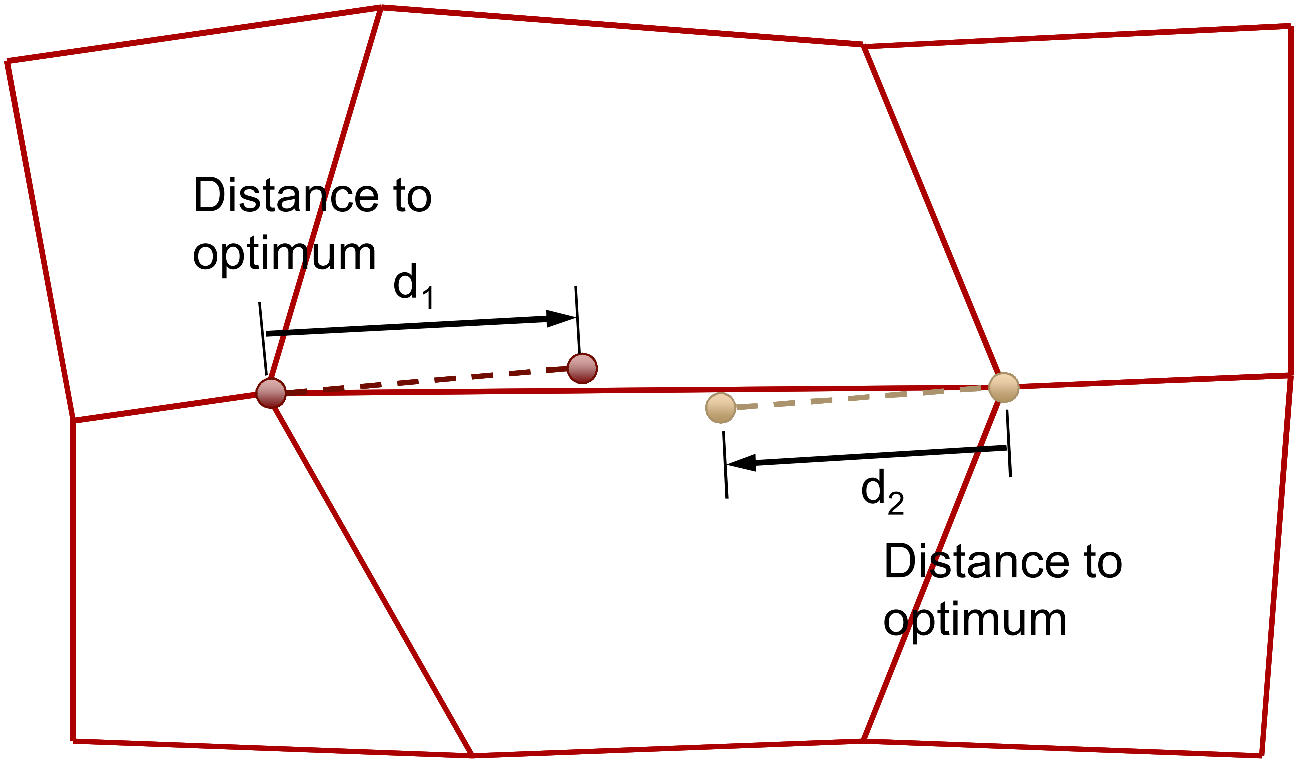
Jacobi Smoothing



Store/update damped location of node

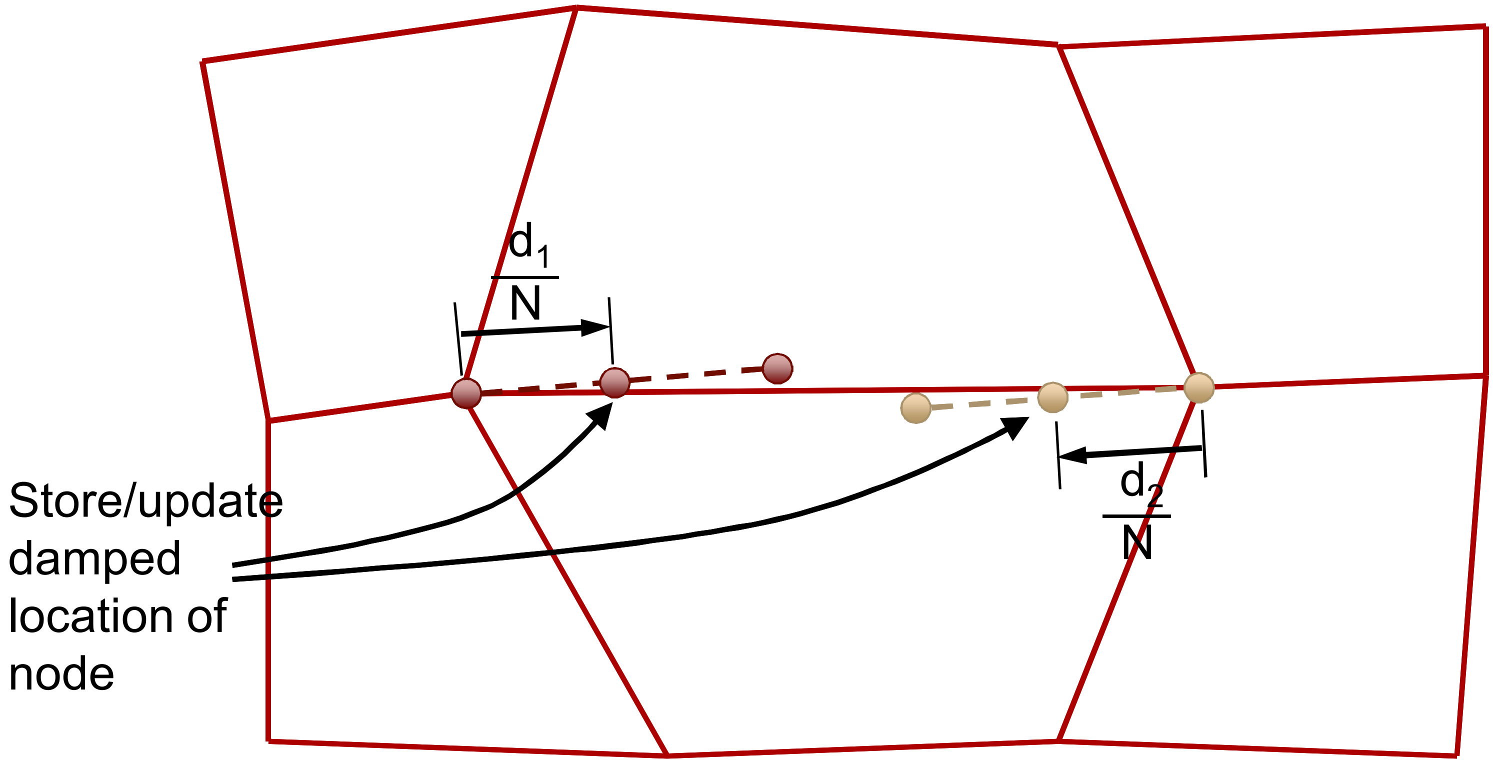
Solution: Apply *damping* to node movement

Jacobi Smoothing



Jacobi Smoothing

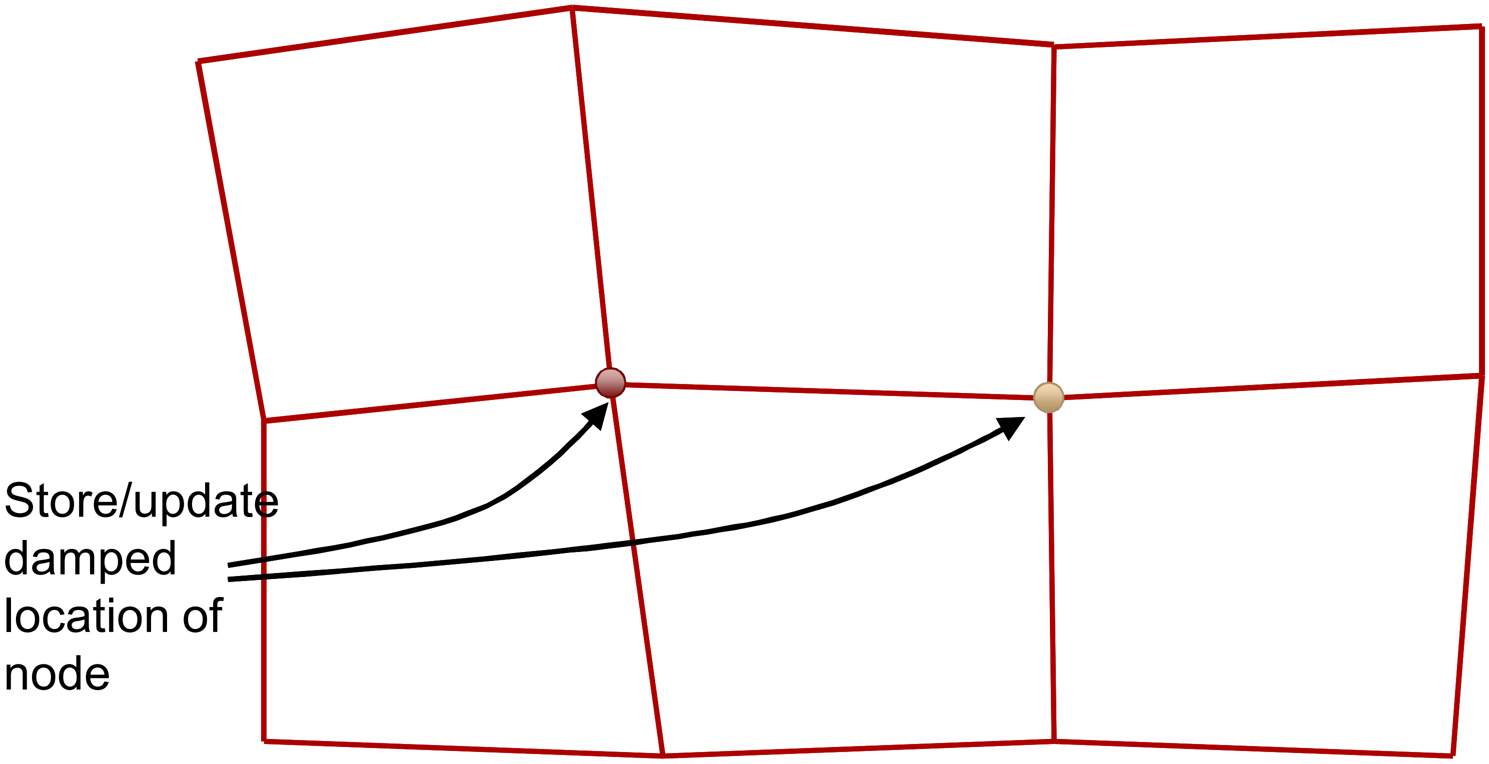
N = number of remaining smoothing iterations



Store/update damped location of node

Solution: Apply *damping* to node movement

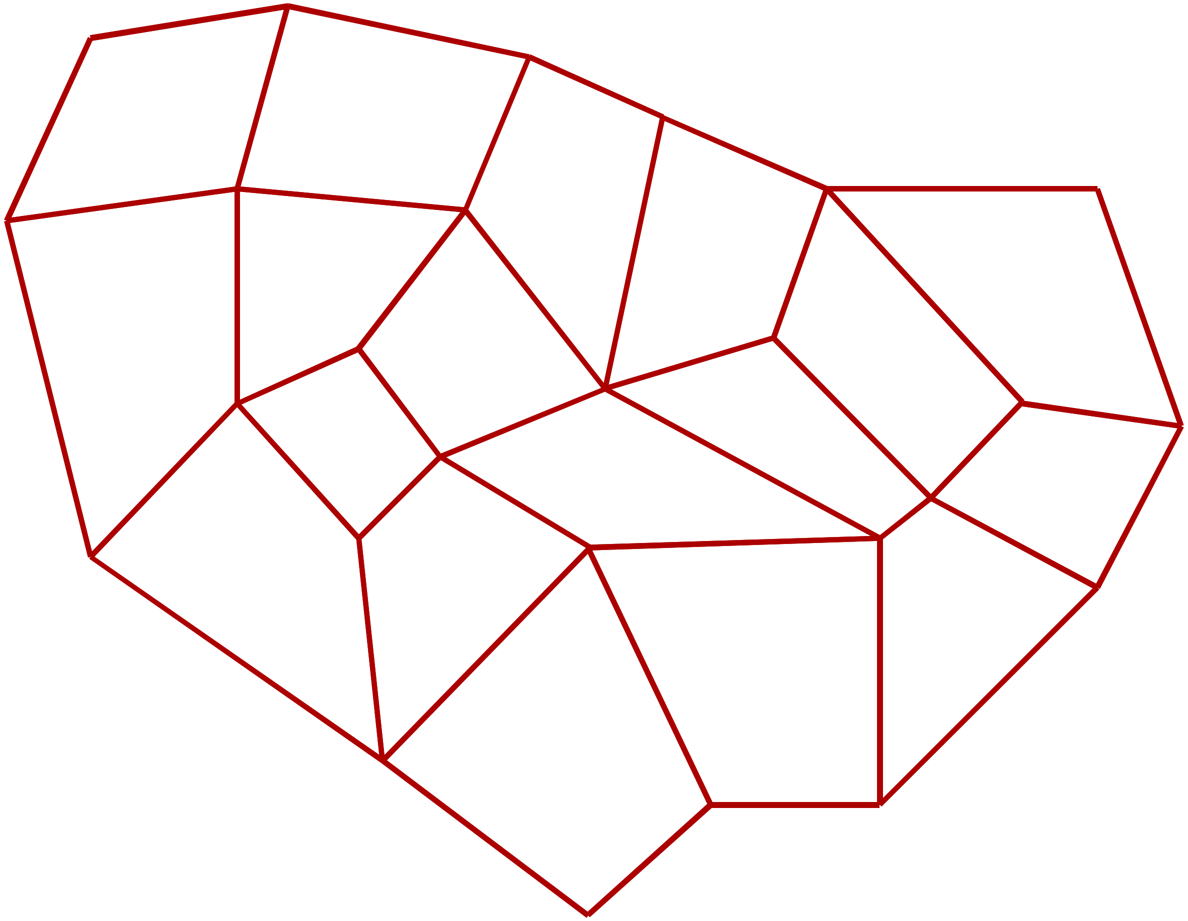
Jacobi Smoothing



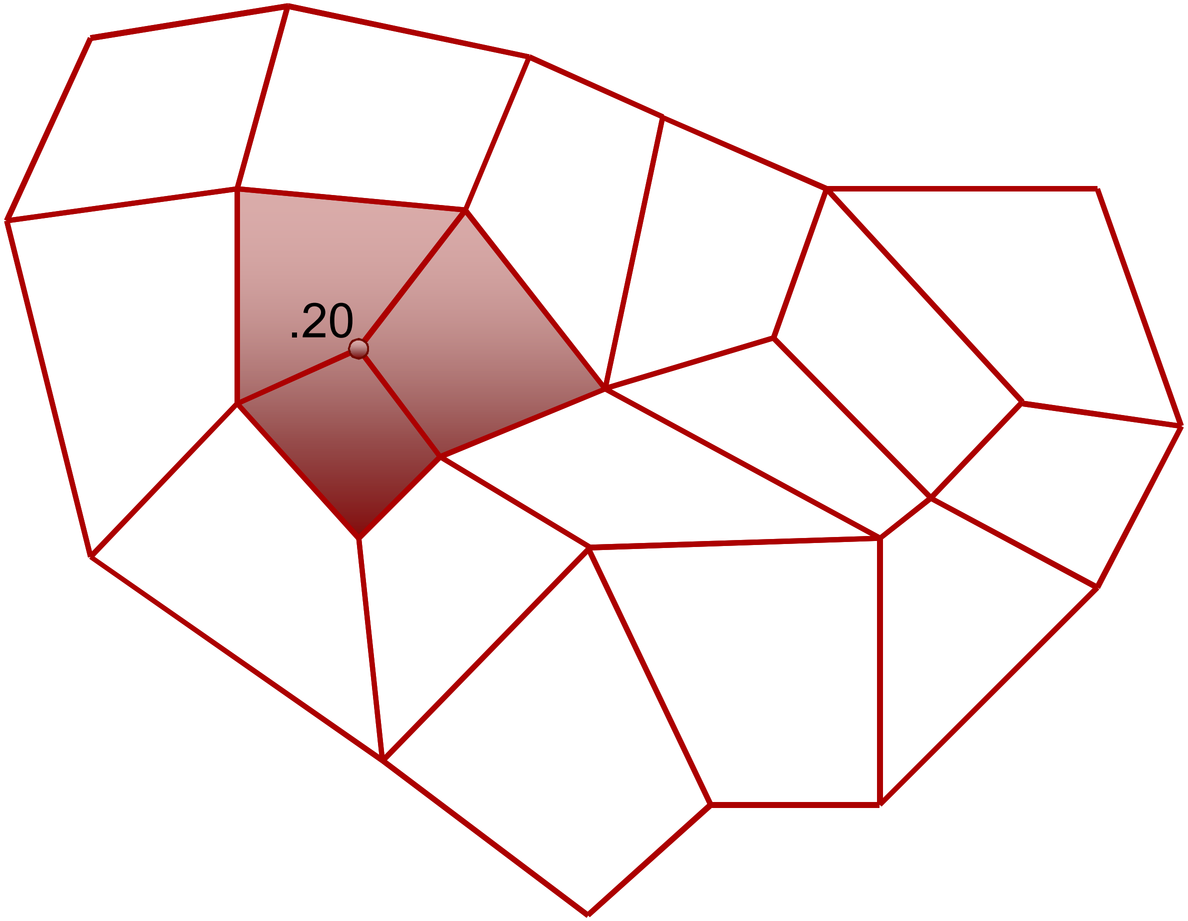
Store/update damped location of node

Solution: Apply *damping* to node movement

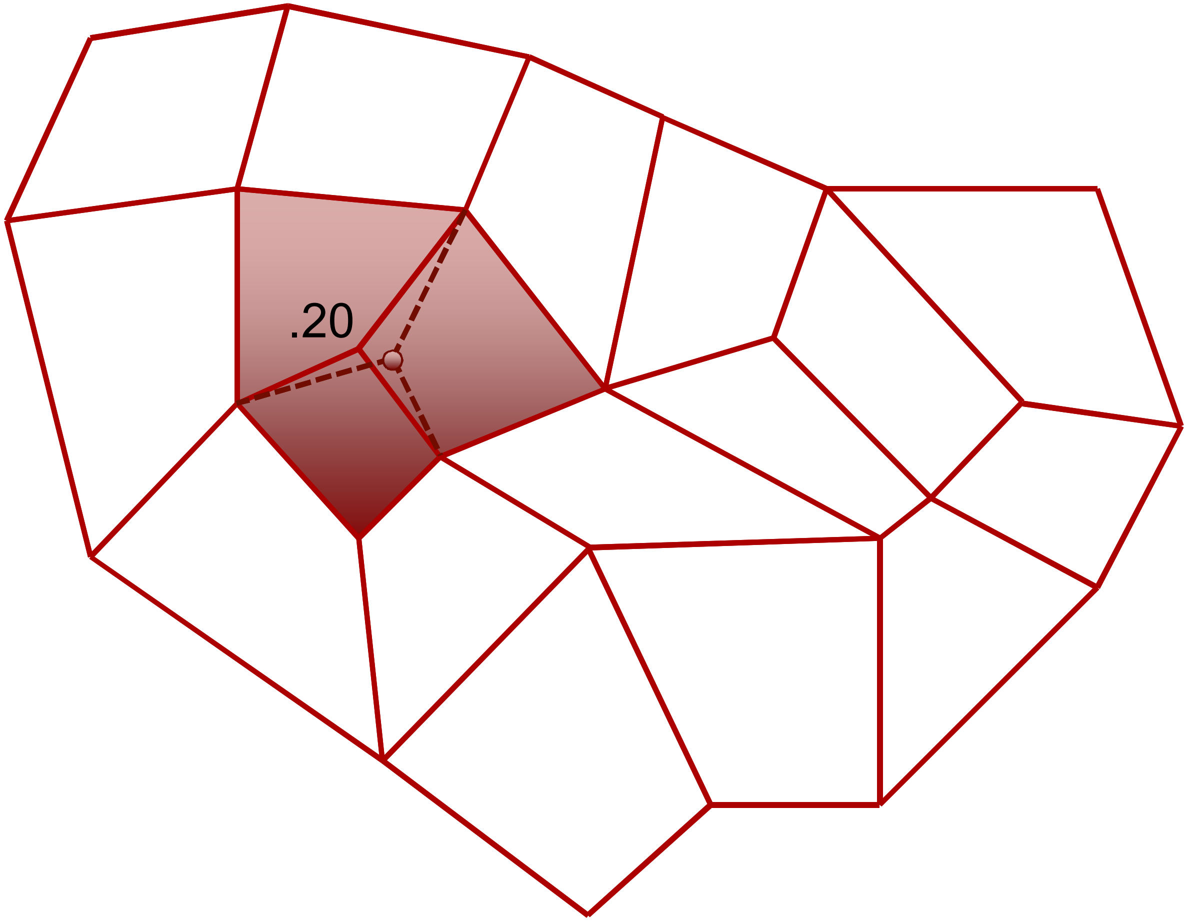
Parallel Coloring



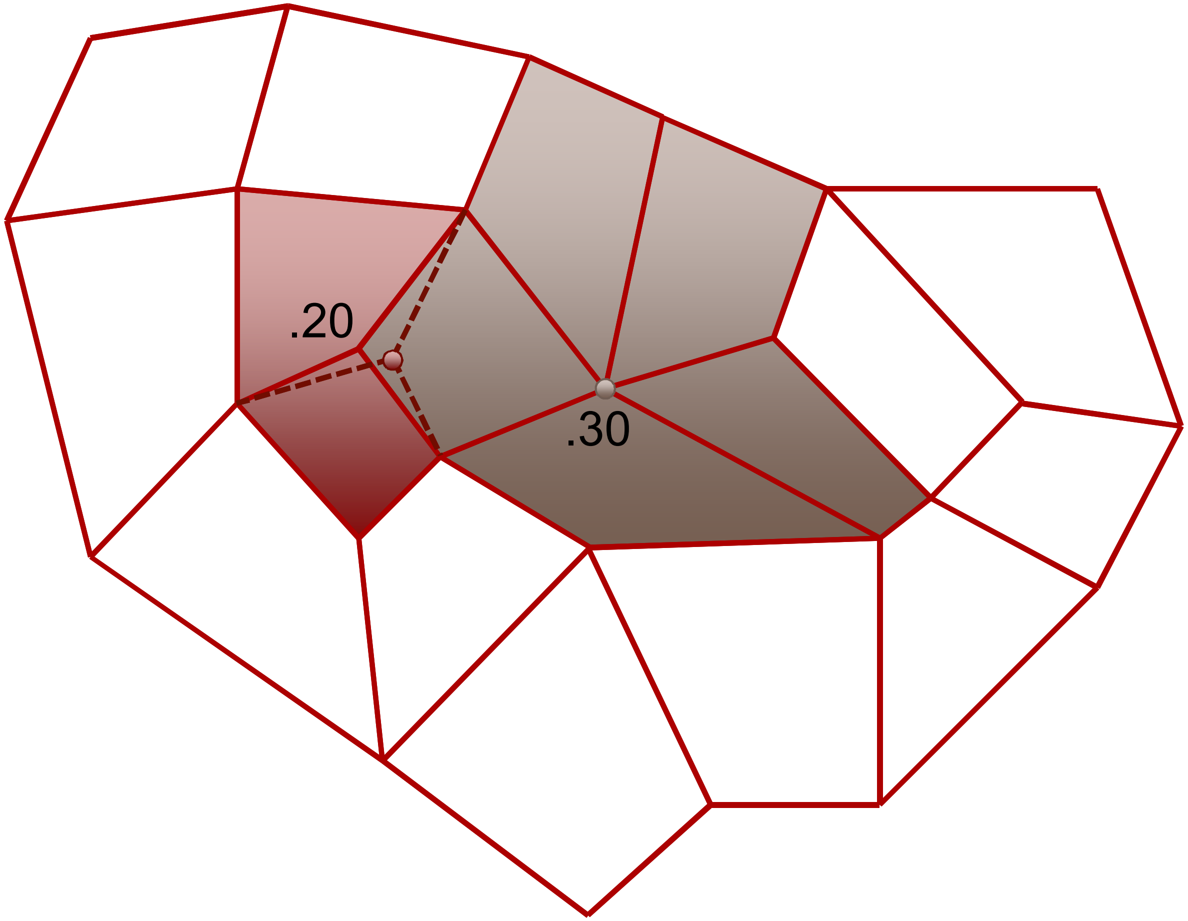
Parallel Coloring



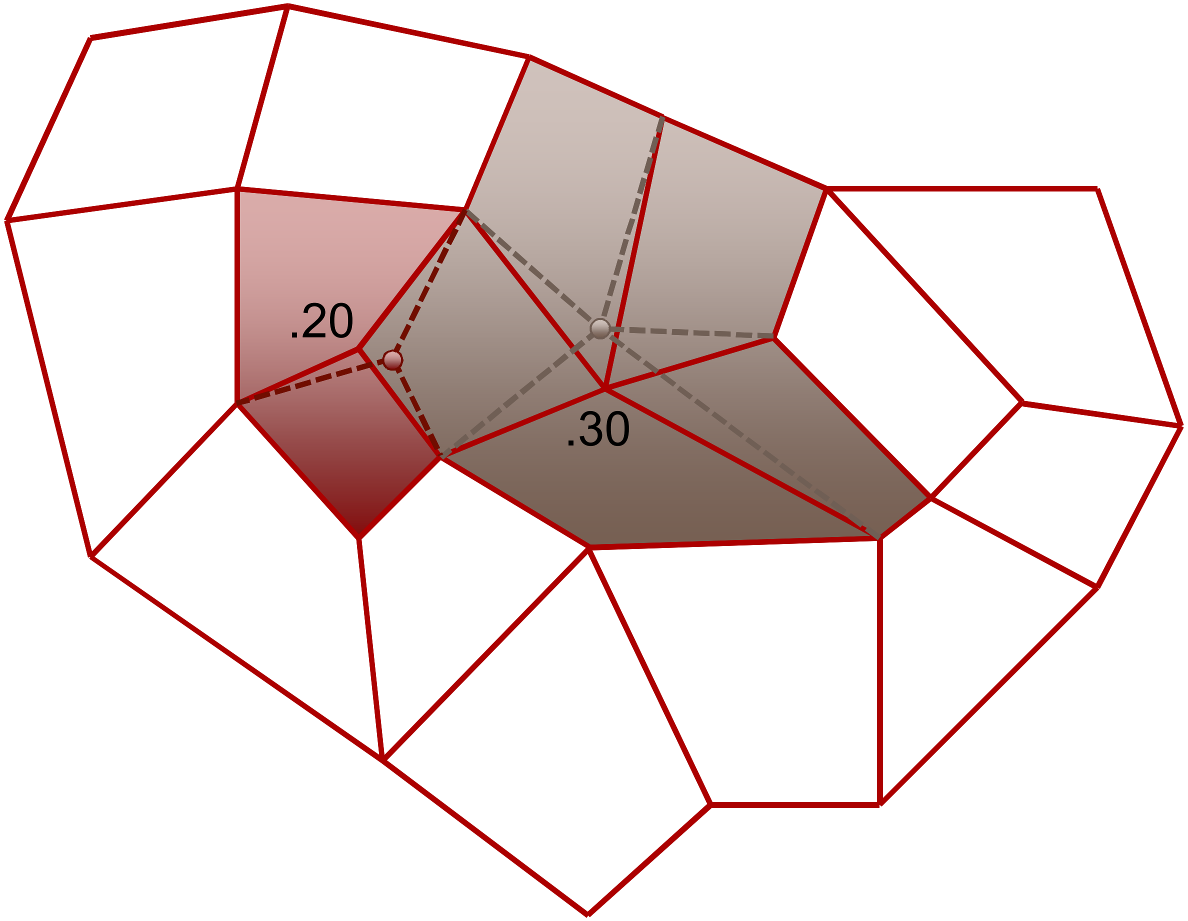
Parallel Coloring



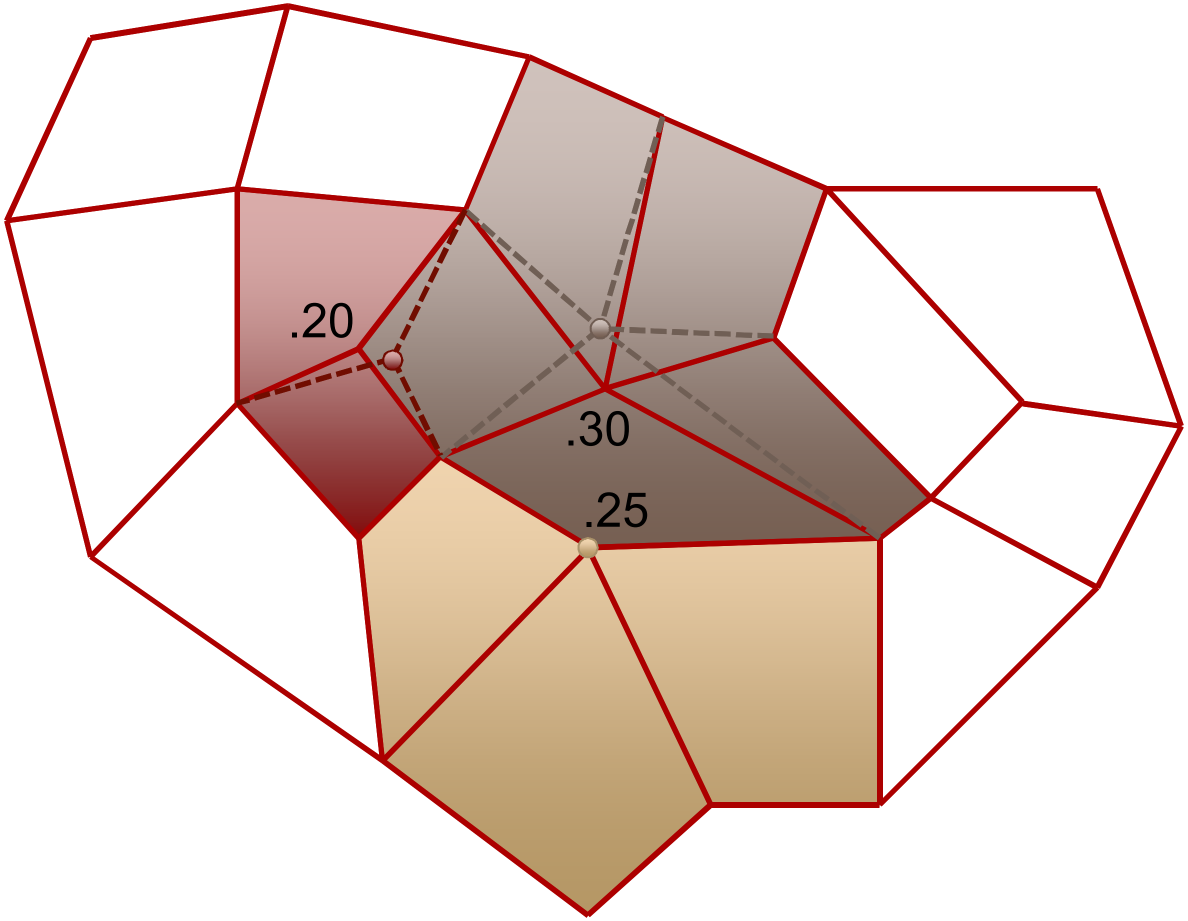
Parallel Coloring



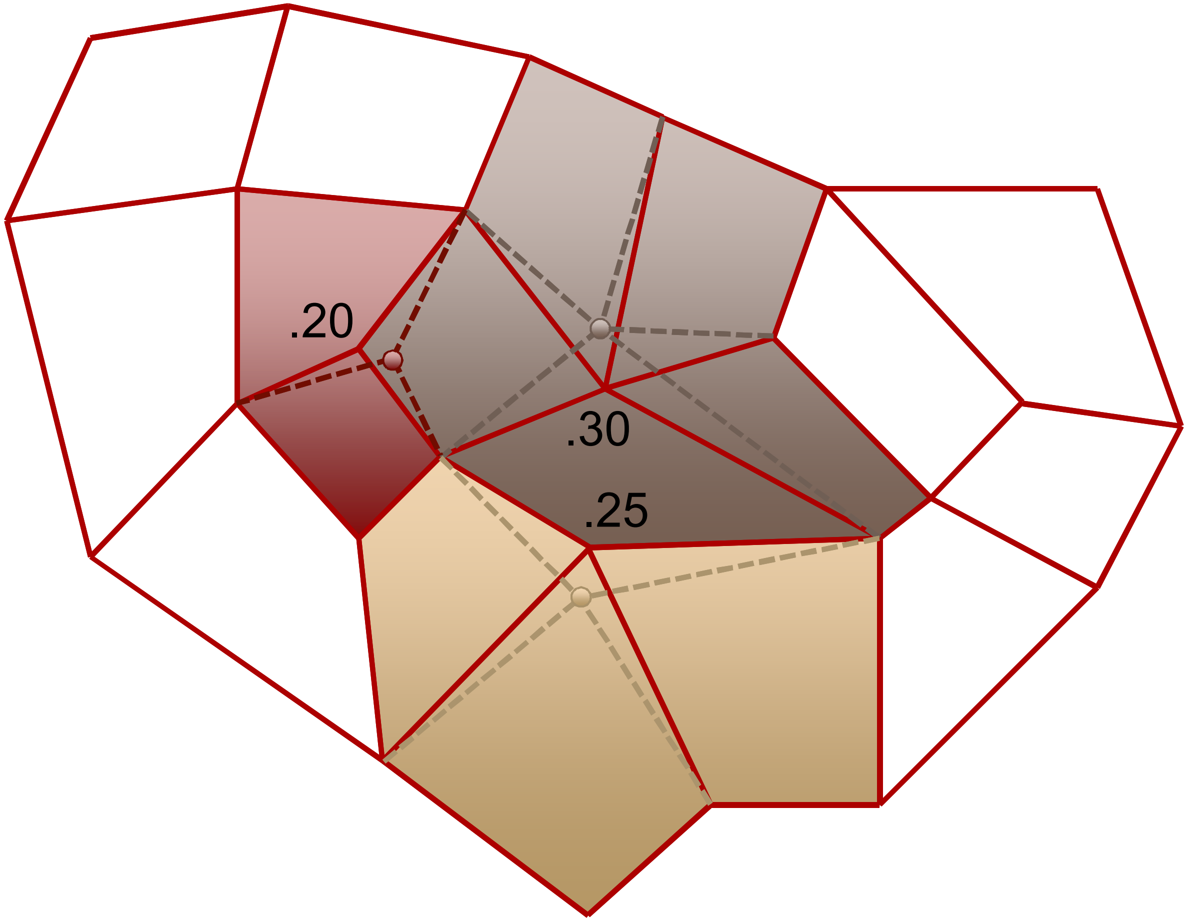
Parallel Coloring



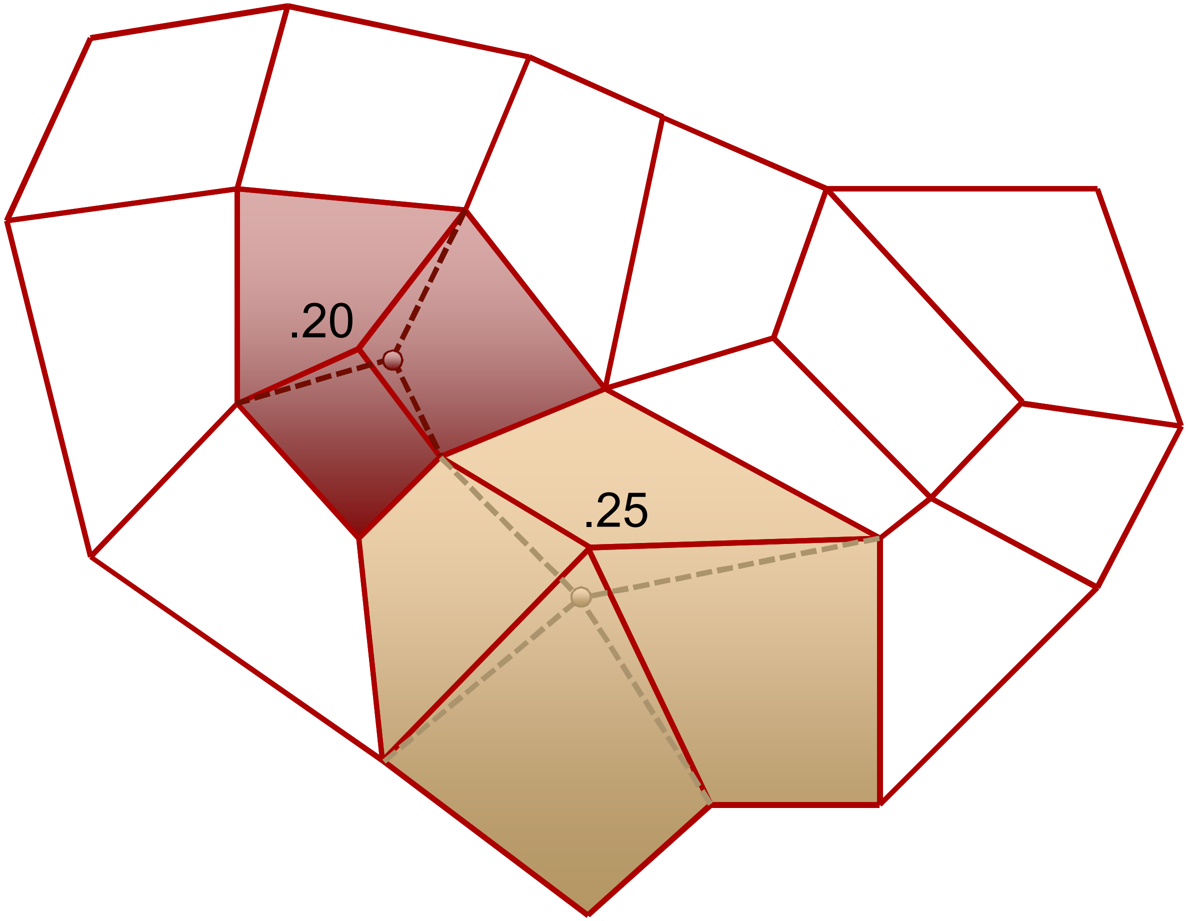
Parallel Coloring



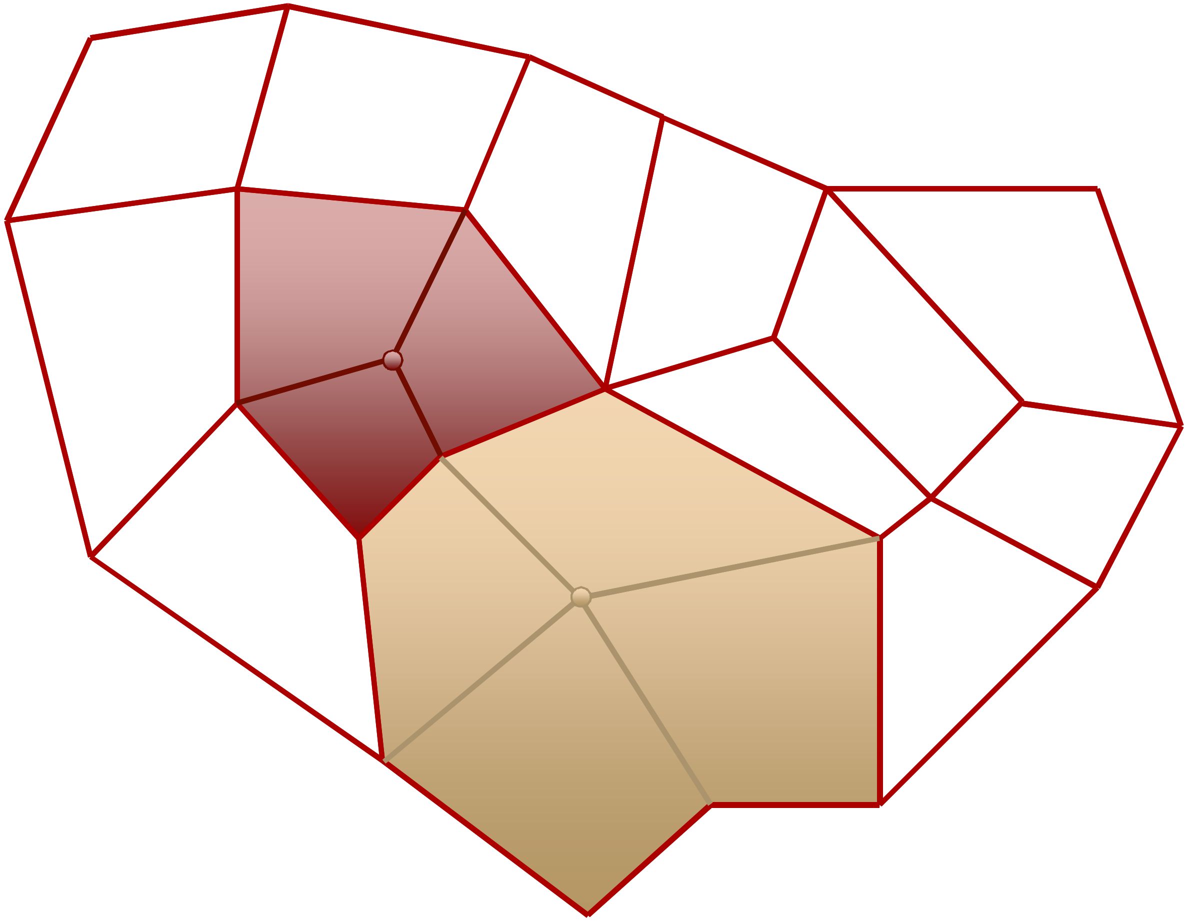
Parallel Coloring



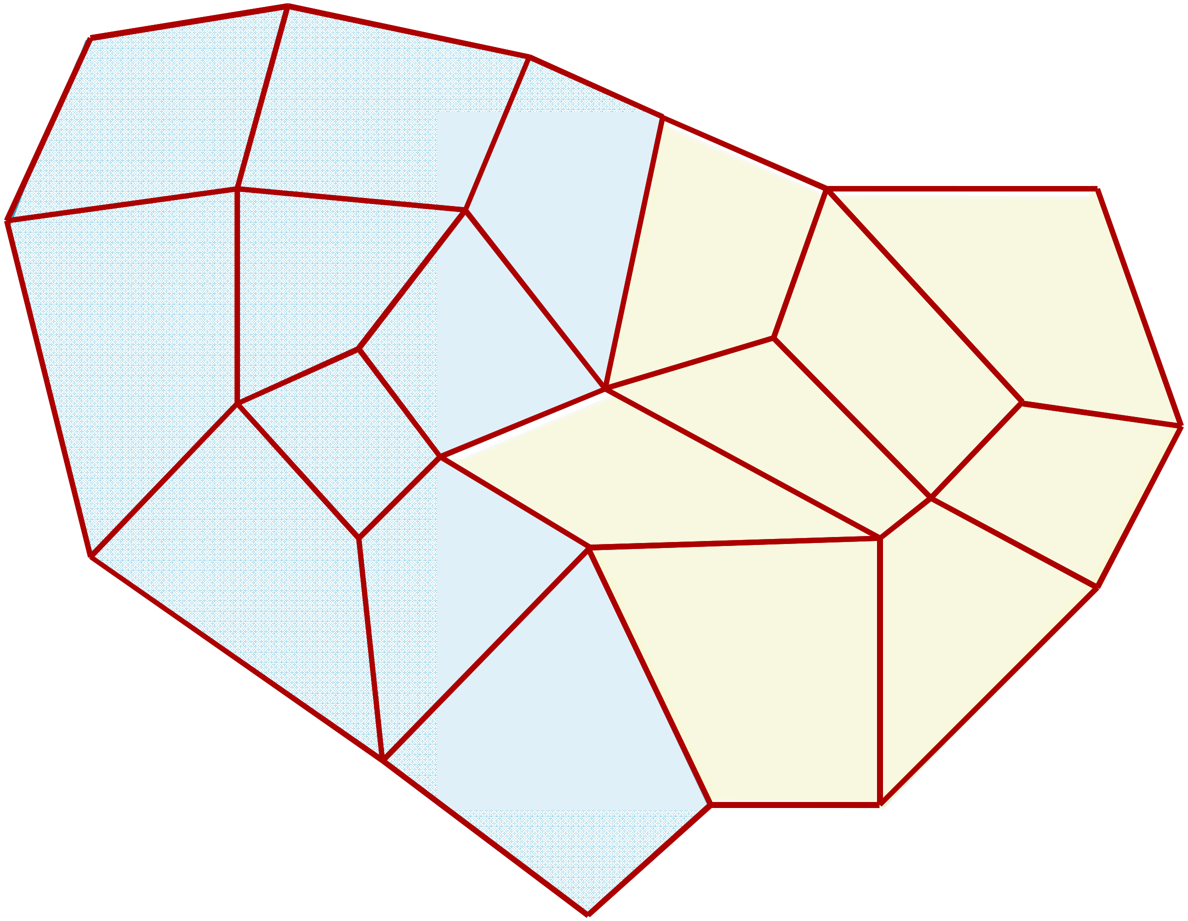
Parallel Coloring



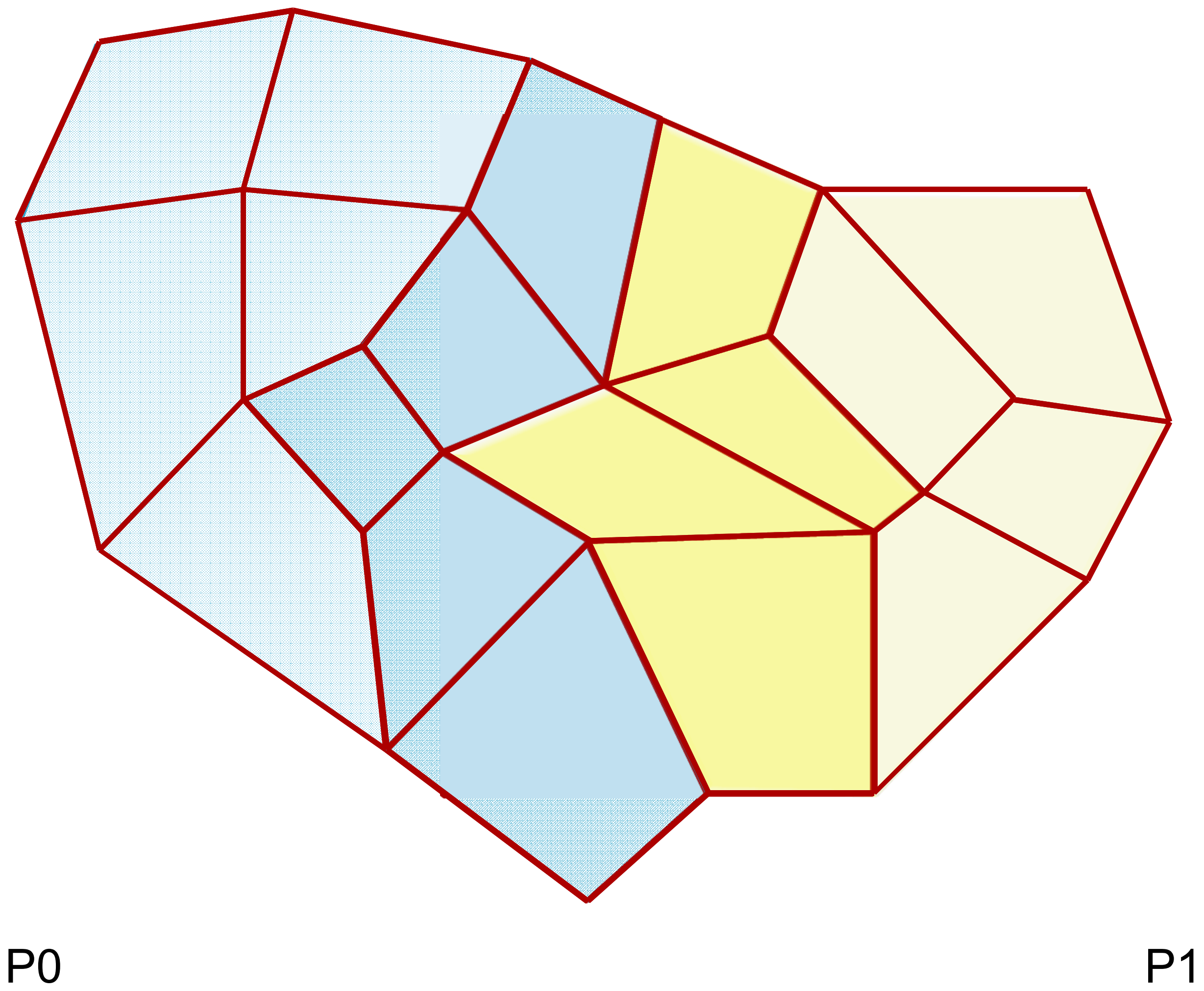
Parallel Coloring



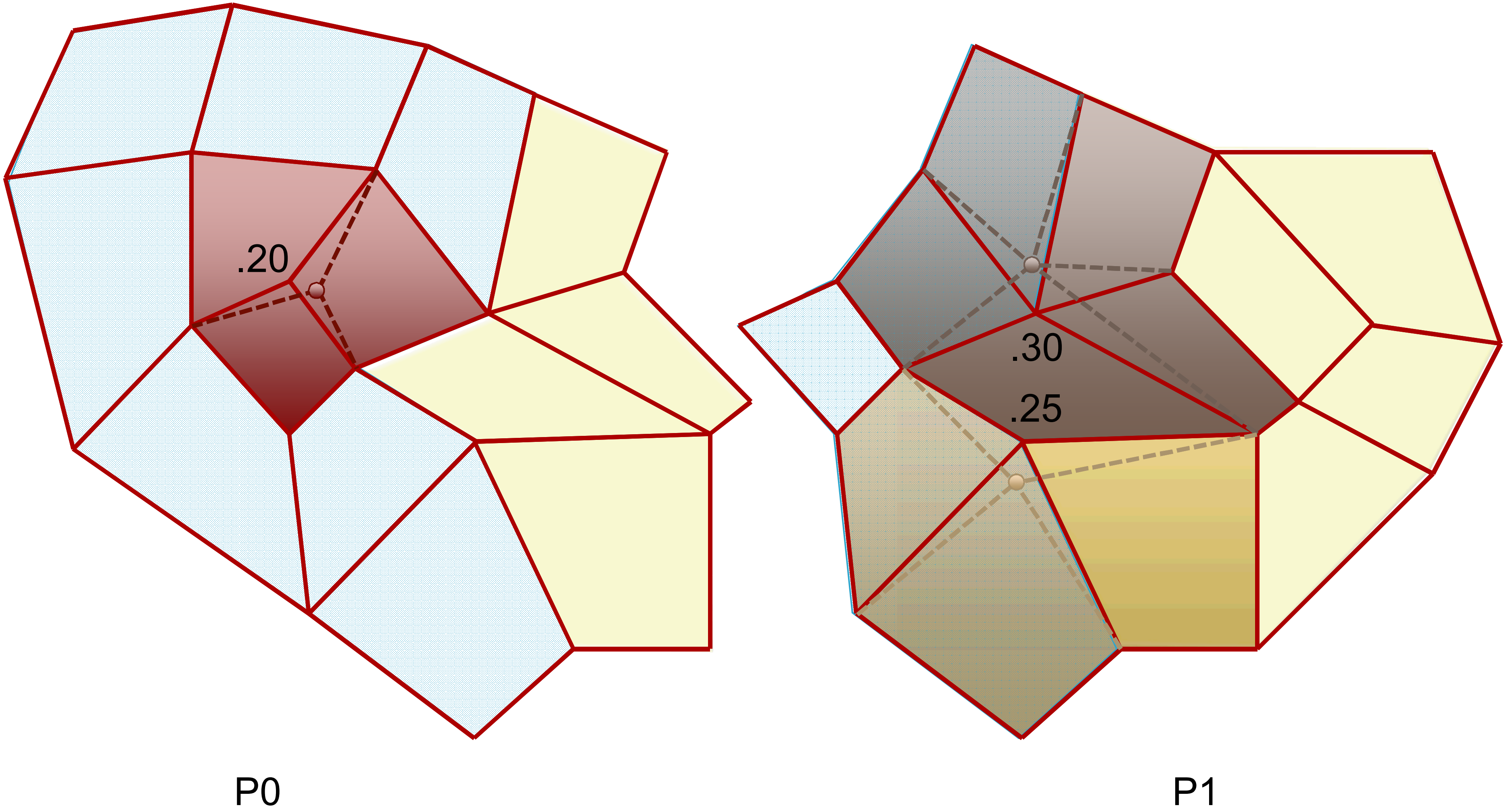
Parallel Coloring



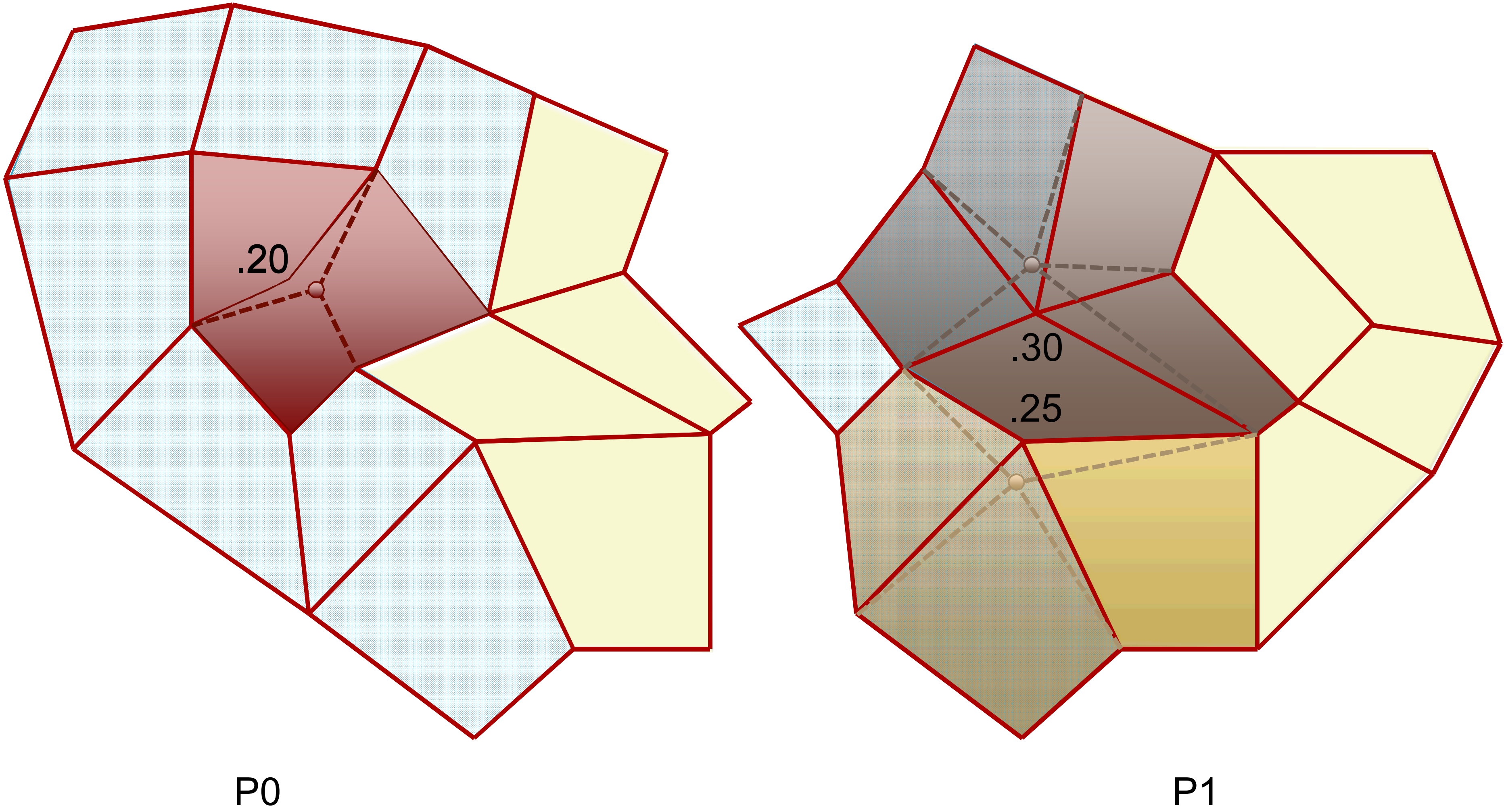
Parallel Coloring



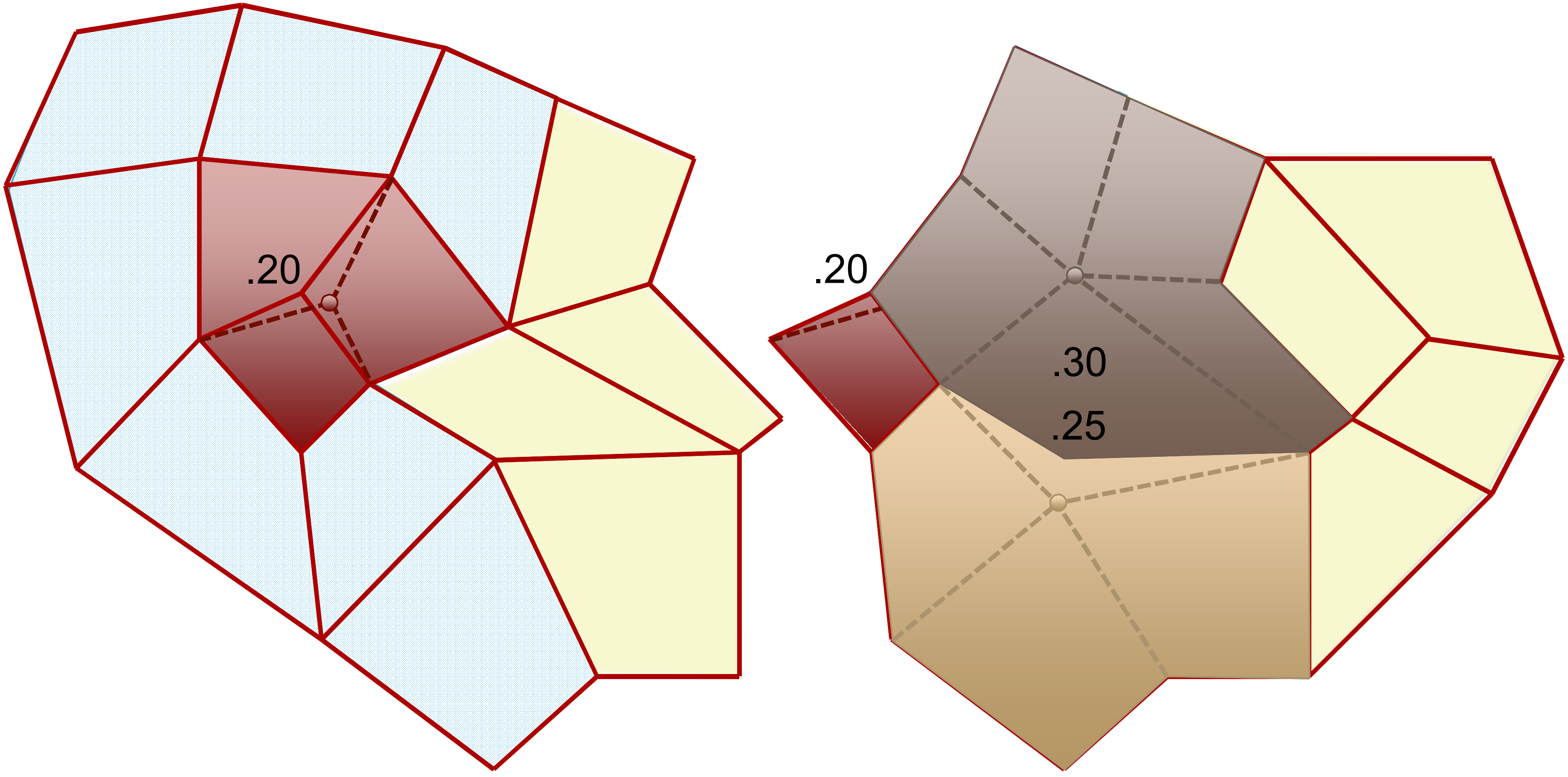
Parallel Coloring



Parallel Coloring



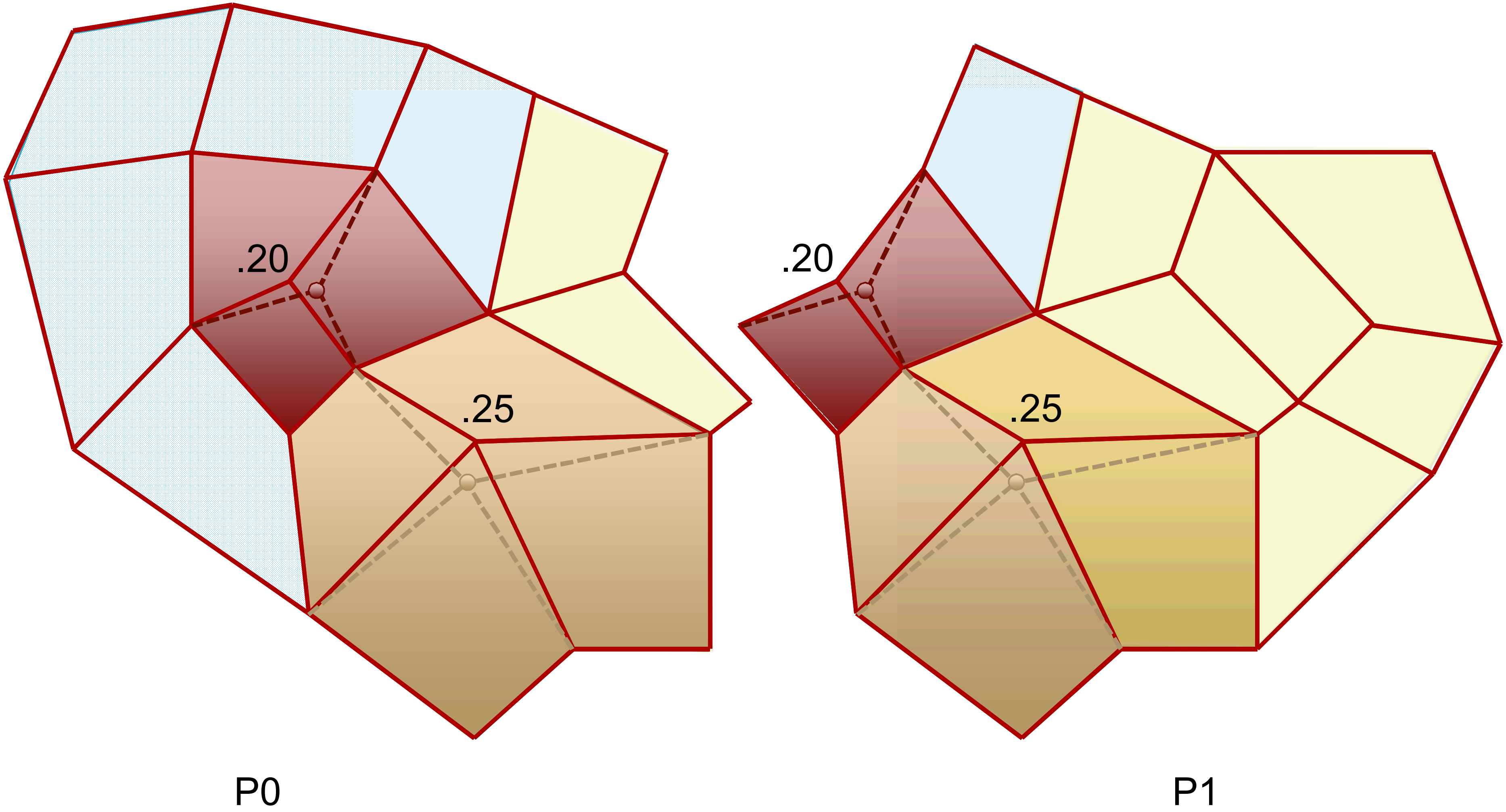
Parallel Coloring



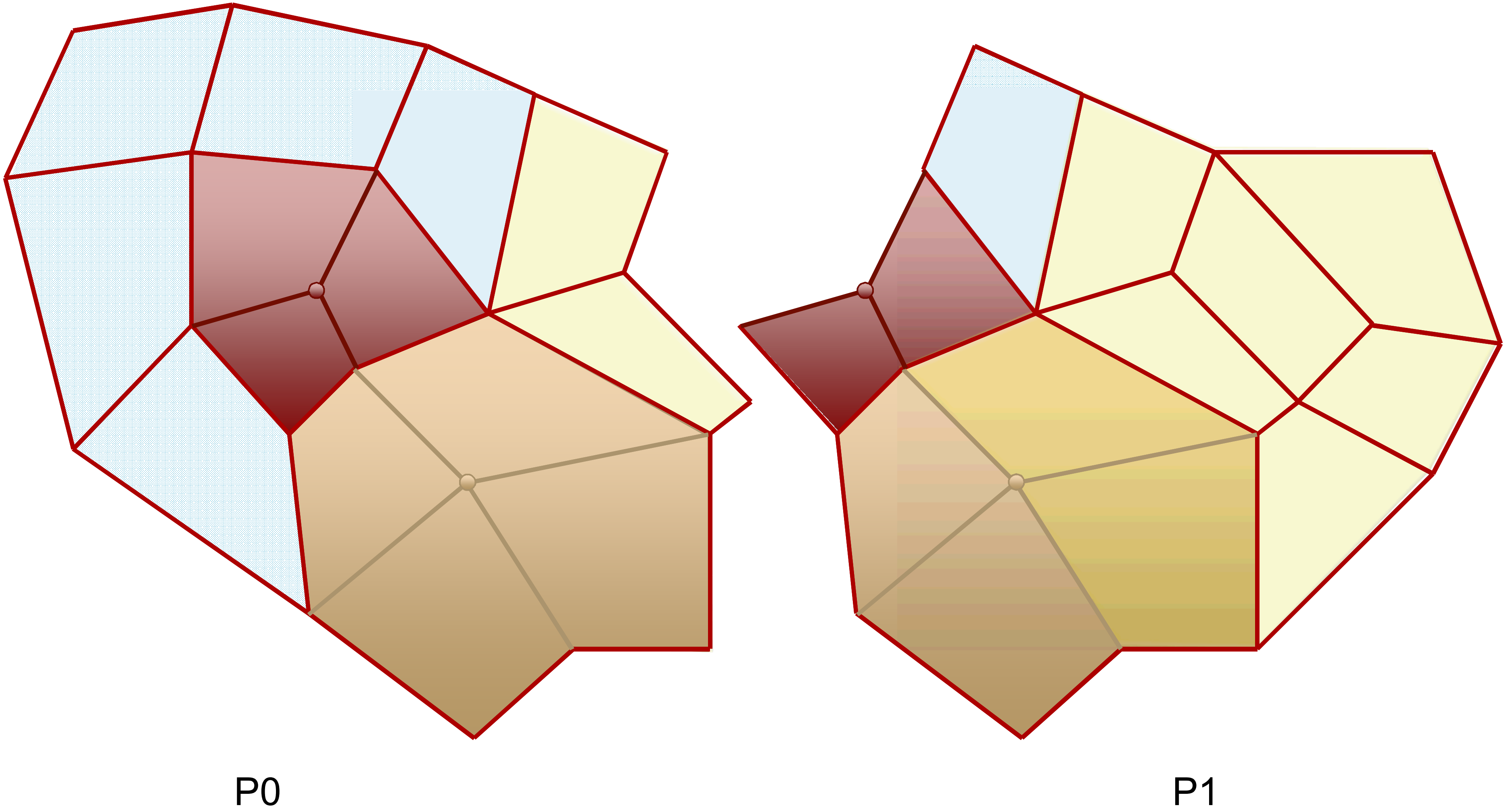
P0

P1

Parallel Coloring

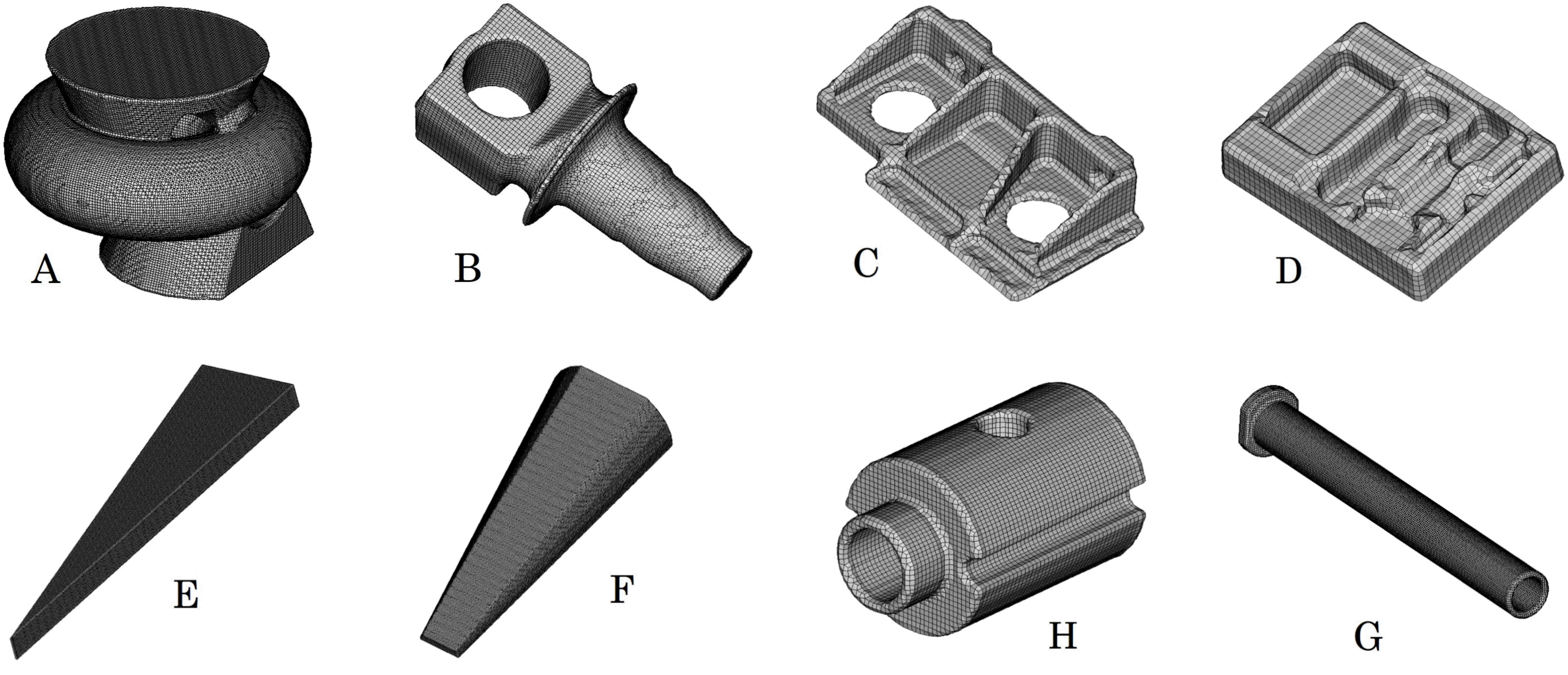


Parallel Coloring



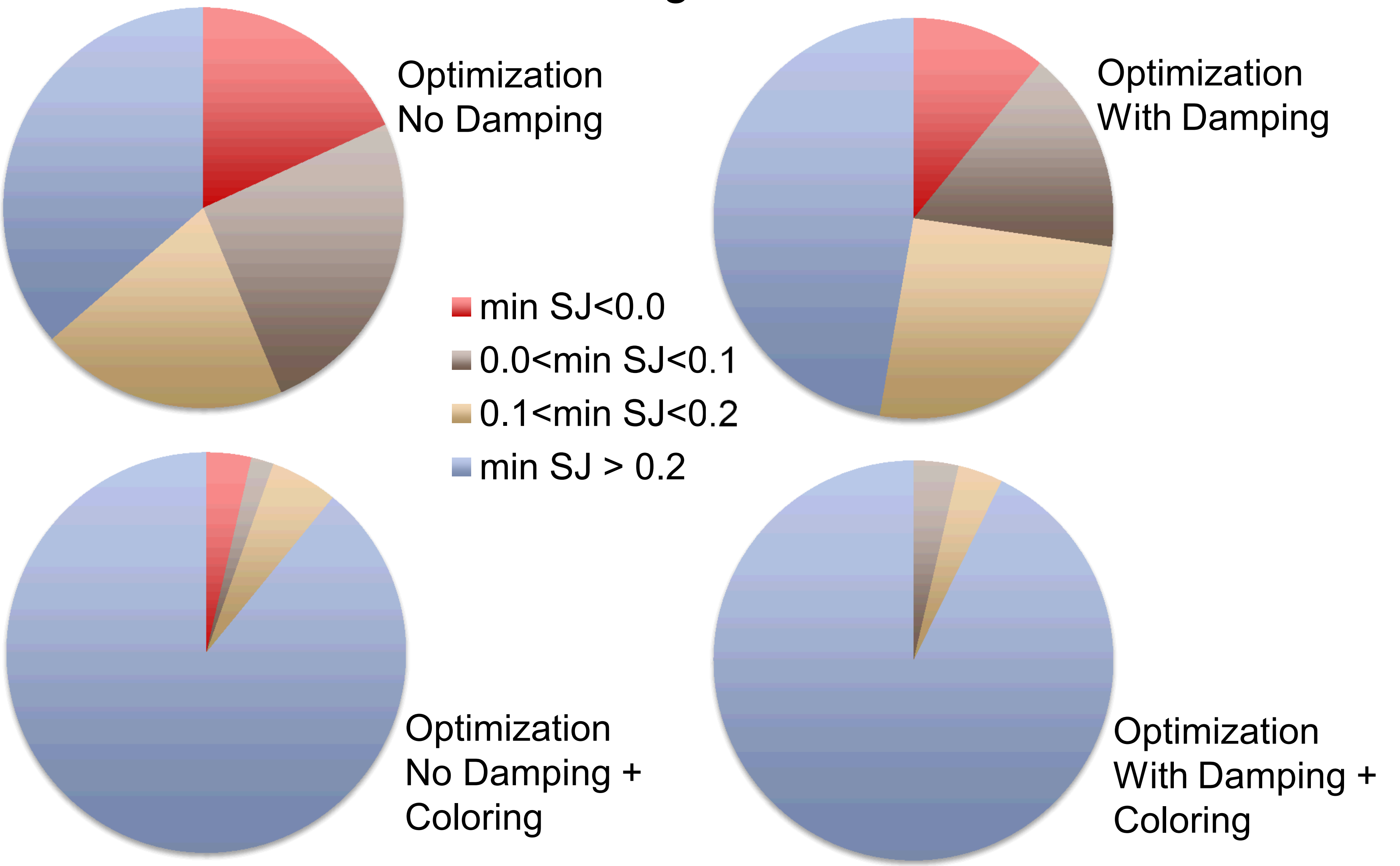
Sculpt Smoothing Comparison

Test Suite: 52 Single Part CAD Models



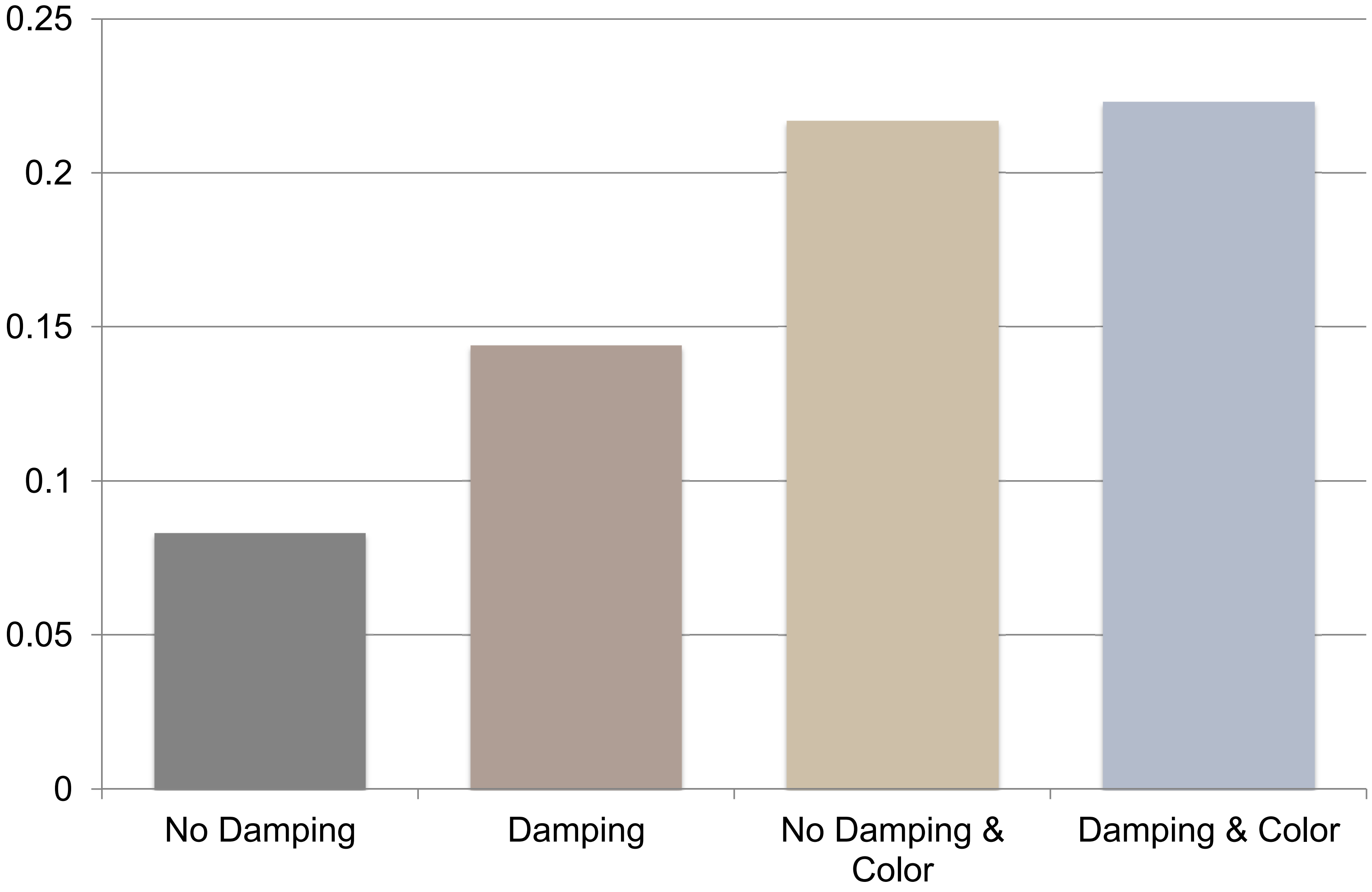
Minimum Mesh Quality

Test Suite: 52 Single Part CAD Models



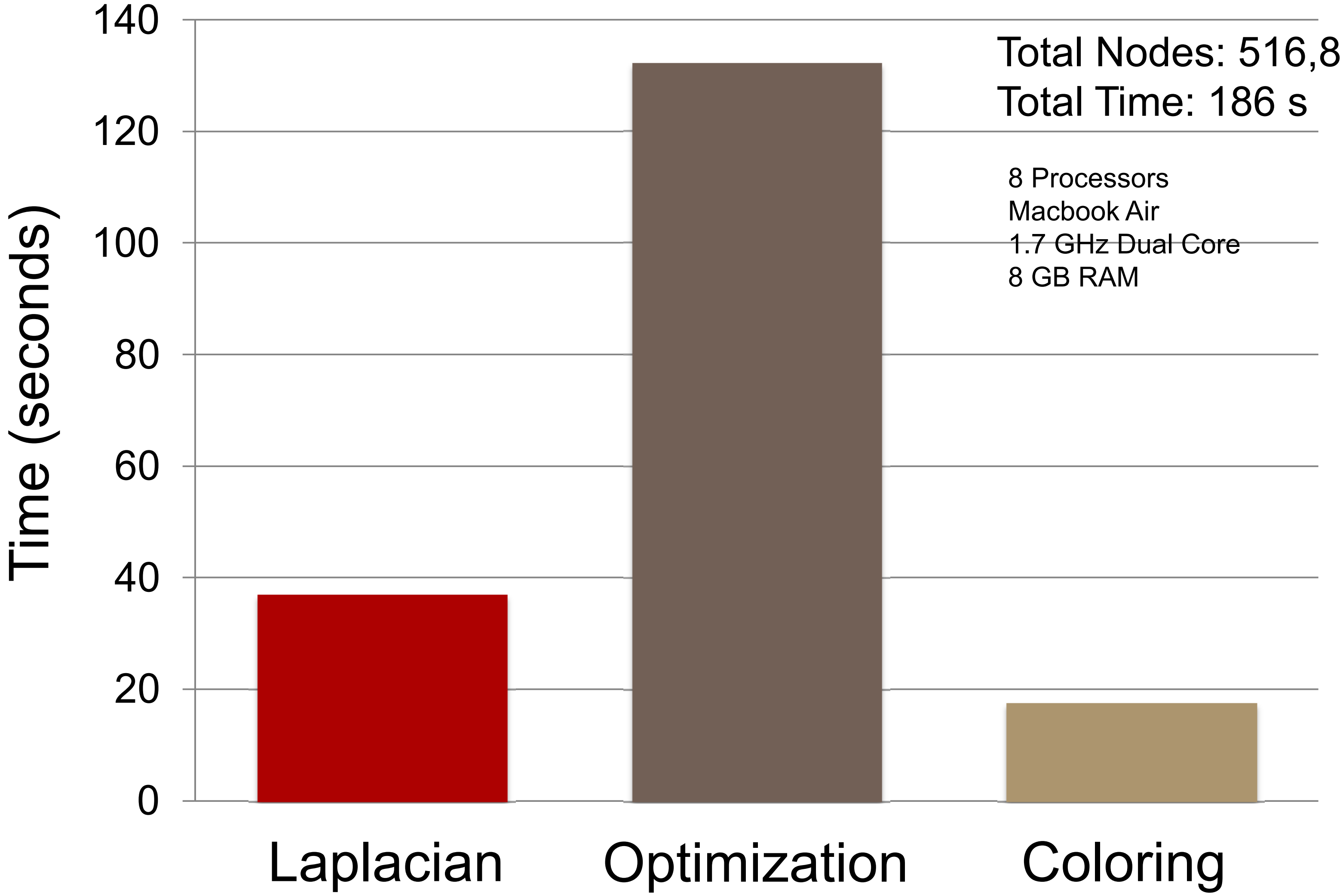
Average Minimum Mesh Quality

Test Suite: 52 Single Part CAD Models



Total Time in Smoothing

Test Suite: 52 Single Part CAD Models



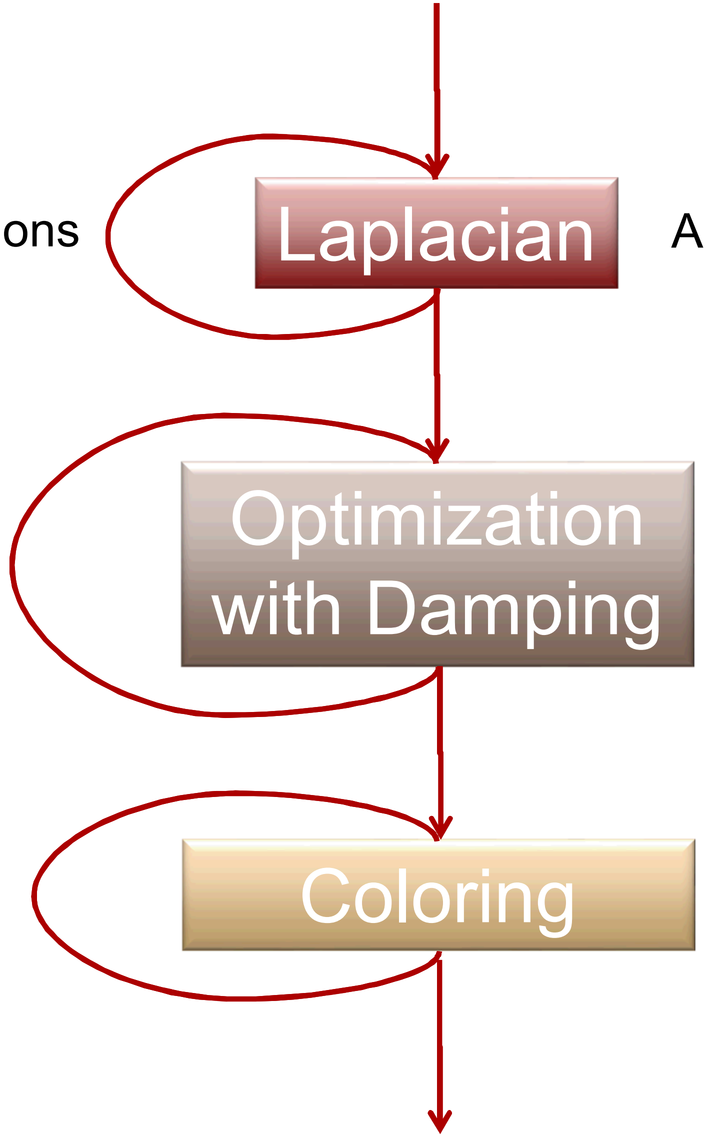
Total Nodes: 516,806
Total Time: 186 s

8 Processors
Macbook Air
1.7 GHz Dual Core
8 GB RAM

Smoothing Procedure

Default iterations

Applied to which elements



2 iterations

Laplacian

All Hexes

5 iterations

Optimization with Damping

Hexes with $S_J < 0.6$

Up to 100 iterations

Coloring

Hexes with $S_J < 0.2$