Parallel Tetrahedral Mesh Refinement

Mehmet Balman

Computer Engineering,
Boğaziçi University

Adaptive Mesh Refinement (AMR)

A computation technique used to improve the efficiency of numerical systems for Partial Differential Equations.

The main idea is to refine regions in which higher resolution is required.

Used fields:

- Finite Elements Methods
- Computational Geometry
- Numerical Applications
- Computer Graphics
- Solid Modeling

Problems & Limitations

Problem size and computational cost grow very rapidly in *3*-dimensional refinement algorithms.

For complex *3-D* problems, the application of the refinement method may not be feasible due to the large matrices involved and their high storage requirements.

Thesis Statement

The overall mesh structure can be partitioned in order to fit into the local memory of computational nodes, and the Rivara's longest edge bisection technique can be used to process the refinement operation locally;

results in each node can be synchronized in a proper and efficient way using appropriate data structure to handle the refinement process in a parallel manner,

in which the resulting mesh data is parallely constructed.

Roadmap

- Mesh Refinement
    - Longest Edge Bisection
    - 4-Triangle and 8-Tetrahedron Algorithms
    - Longest Edge Propagation Path
- Parallel Algorithm
- Test Results
- Conclusion & Contributions

Two Approaches for the Refinement Problem

The longest-edge bisection:

- The longest-edge of the mesh element is always bisected initially.
- A bisected edge influences the neighbor elements and triggers them to be refined.
- guarantees a good-quality, conforming mesh structure with linear time complexity.

Delaunay algorithm:

- adding non-vertex points in the circumcenter of the worst triangles of the current structure.

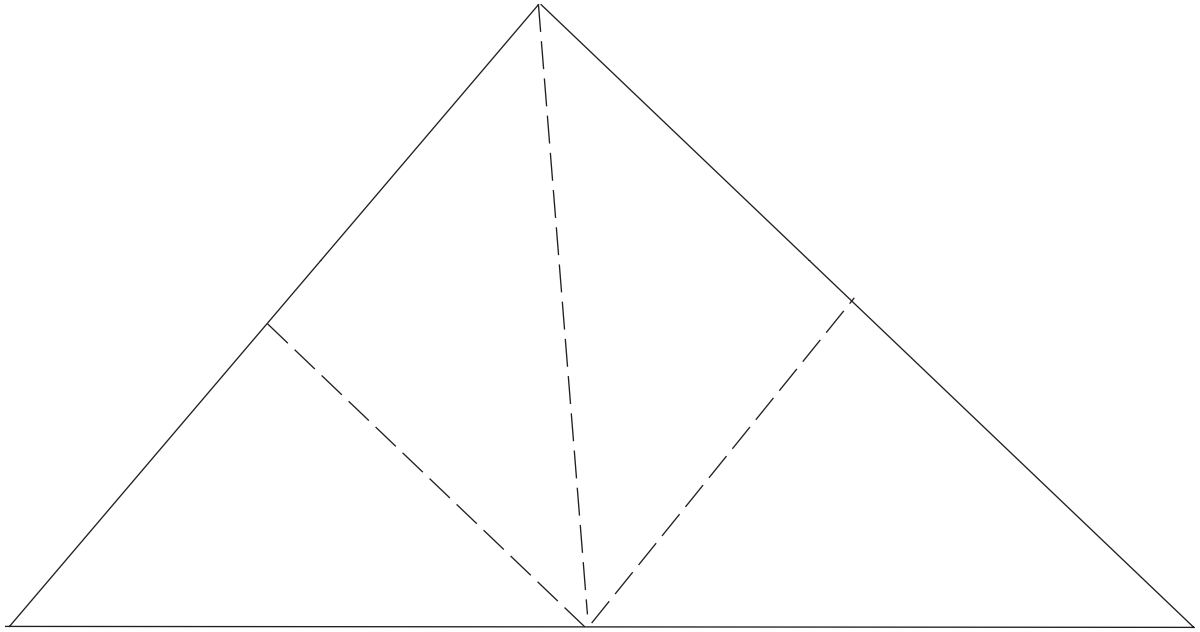
Mesh Conformity

- Intersection of adjacent triangles is either a common vertex or an edge.

- Transition between small and large elements should be gradual.

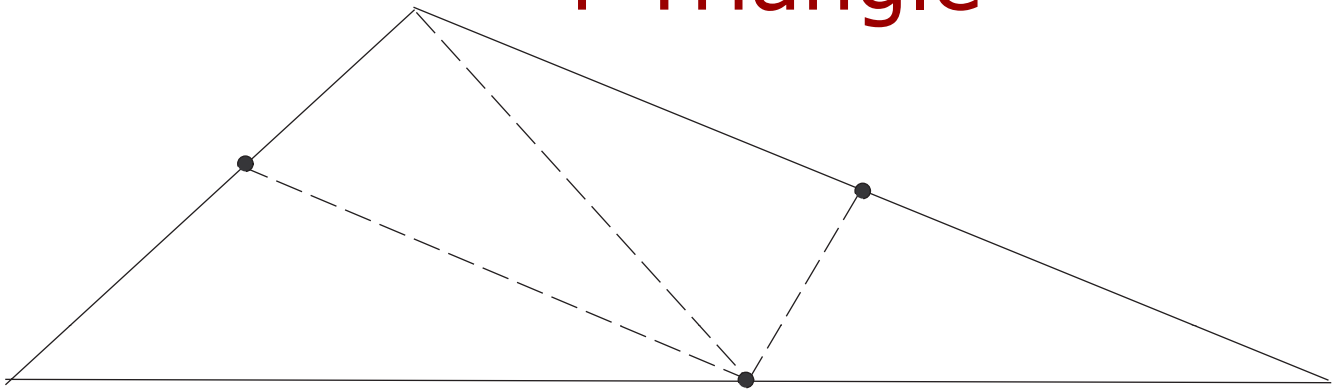
- All angles in triangle refinement are greater or equal to half of the smallest angle in the initial mesh geometry.

4T-LE Refinement

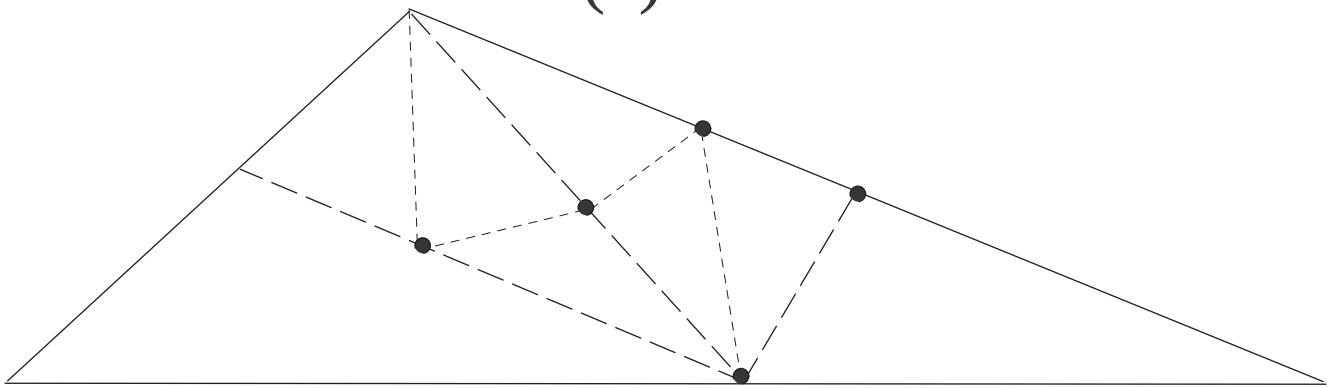


Four Triangle Longest Edge Partition (*4T-LE*) divides a triangle *t* into four sub-triangles such that the original triangle is bisected by its longest edge.

Stable Molecule Behavior of 4-Triangle



(a)



(b)

Longest-Edge Propagation Path (*LEPP*)

For any triangular mesh structure T,

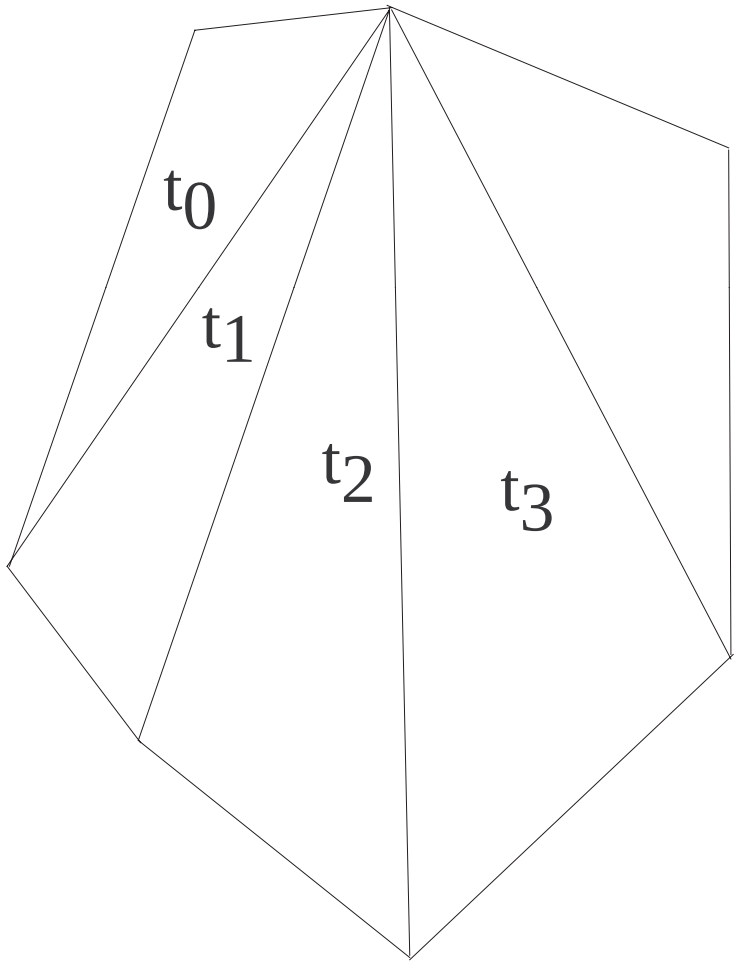
the longest-edge propagation path (*LEPP*) for a triangle t is the ordered list of triangles $\{t_0,t_1,t_2,..\}$, such that t_{i-1} is adjacent to triangle t_{i-1} by the longest-edge (*LE*) of t_{i-1}.

The longest-edge propagation path of triangle t, is always finite and longest-edges in the list of triangles have always increasing lengths.

Longest-Edge Propagation Path (LEPP)

$LEPP(t_0)=\{t_0,t_1,t_2,t_3\}$,

propagation path for triangle t_0



Longest-Edge Bisection Algorithm

Longest-Edge Bisection (*T*, *t*)

While there is a triangle *t* not bisected
compute *LEPP(t)*
if t is the boundary terminal triangle
bisect(t)
else
bisect the last pair of terminal triangles in *LEPP(t)*

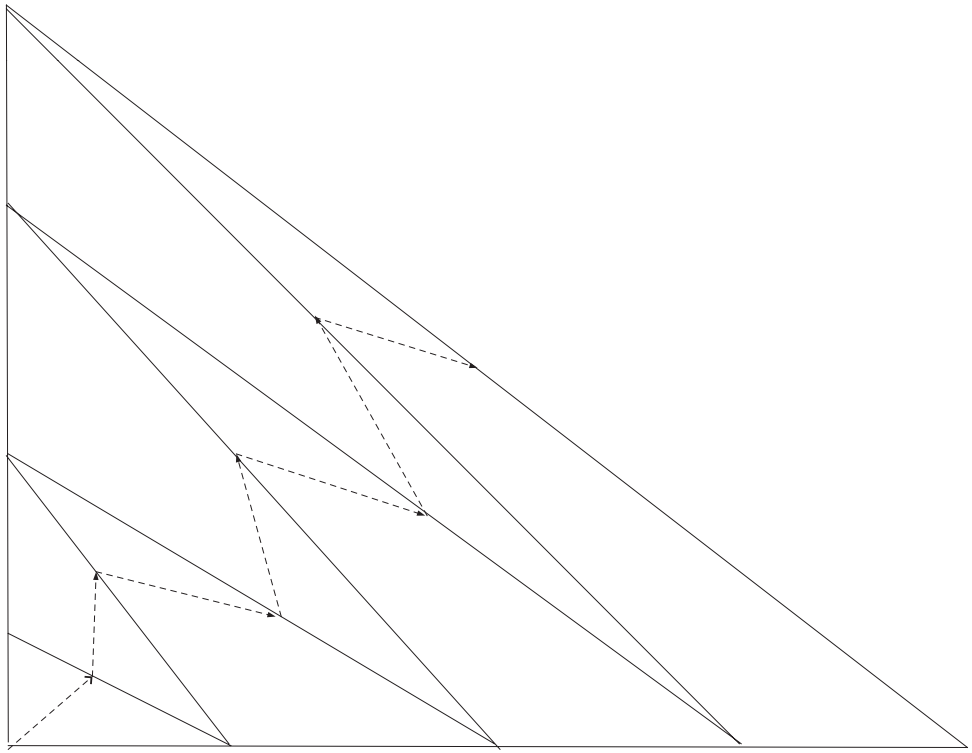
Longest-Edge Bisection

The longest-Edge Propagation Path algorithm can be generalized to *3*-dimensional tetrahedral mesh structures.

LEPP for a tetrahedron t, is the set of neighboring tetrahedra that have adjacent longest-edge greater or equal to the preceding tetrahedra in the list.

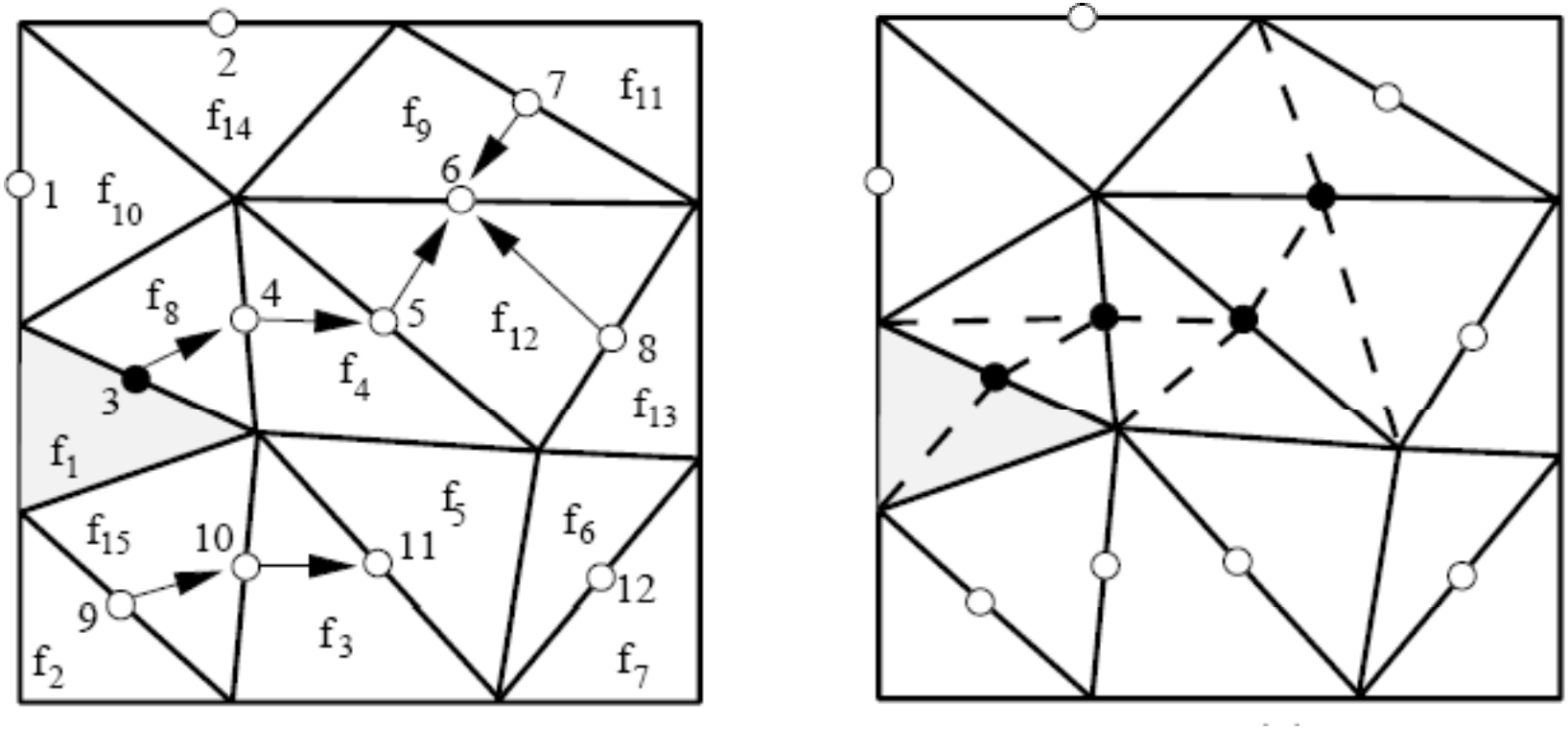
Longest-Edge Propagation

An eccentric situation



According to the theoretical results and experiments, average propagation path is reduced in each refinement stage and approaches to the constant of *5*. (*2-D*)

Propagation of the Refinement (*2-D*)

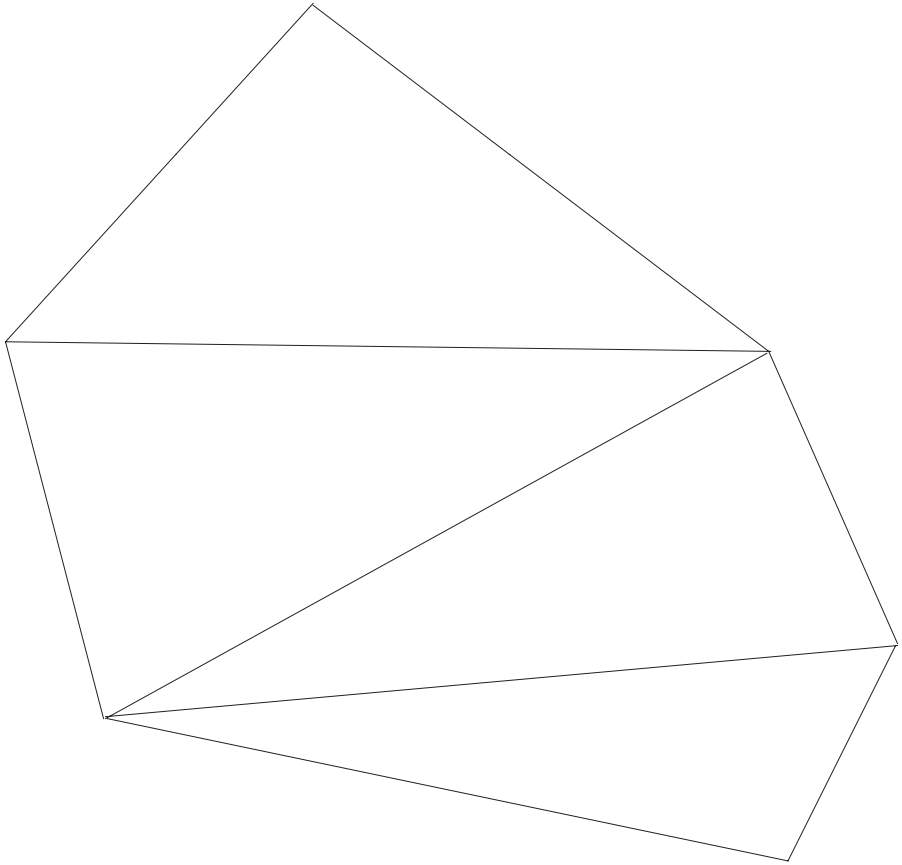


(Ozturan, 1996)

Skeleton Algorithms

The 2-dimensional view of a tetrahedron.

Volume refinement is based on the partitioned triangular faces of the tetrahedron in 2-dimension



Skeleton Algorithms

The *3*-dimensional skeleton algorithm is a generalized version of 4-Triangles.

If tetrahedral mesh τ is conforming, then *2-D* skeleton, which is the triangular faces of the elements of τ, is also conforming.

Moreover, a *1*-D skeleton of τ tetrahedral mesh is a conforming wireframe mesh of the elements of τ.

Skeleton Algorithm (*3-D*)

Partition edges over *1-D* skeleton.

Partition faces over *2-D* skeleton that are involved in the refinement.

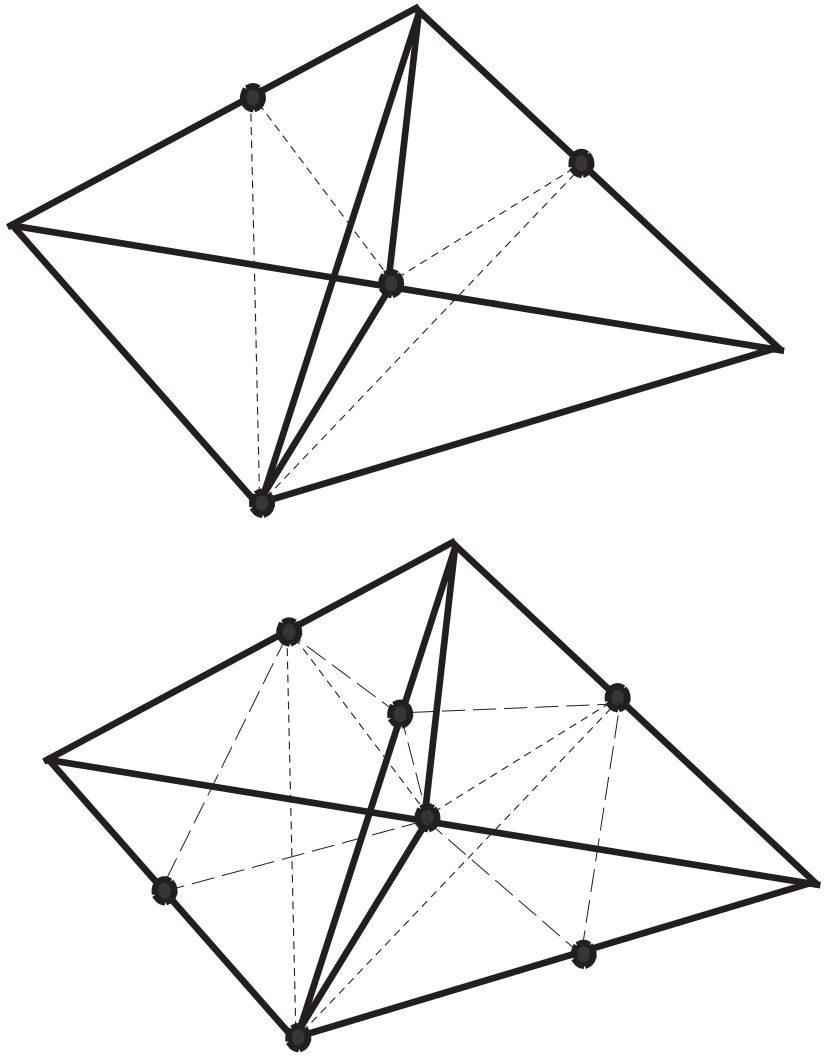
(*4-Triangle LE*)

Partition involved tetrahedrons according to partition patterns.

(*8-Tetrahedra LE*)

4-Tetrahedra & 8-Tetrahedra

The 8-Tetrahedra longest edge partition is a *3-D* algorithm that can be explained by applying 4-Triangle skeleton refinement methodology to the faces of corresponding mesh τ.



8T-LE

Information in the lower dimension is used to partition appropriate triangles.

The 8-tetrahedra algorithm is the generalization of skeleton algorithms in 3-dimensions.

Volume triangulation with 8 new internal tetrahedra occurred and each face has been partitioned into 4 triangles.

Such a triangulation produces conformity both in volume and surface structure.

Longest Edge Propagation Path

In *2-D* meshes, a *LEPP-graph* forms a forest

each edge can only have two neighbor triangles.

For *3-D* meshes, a directed-acyclic graph (*DAG*) is formed

each longest edge can be shared by many triangles

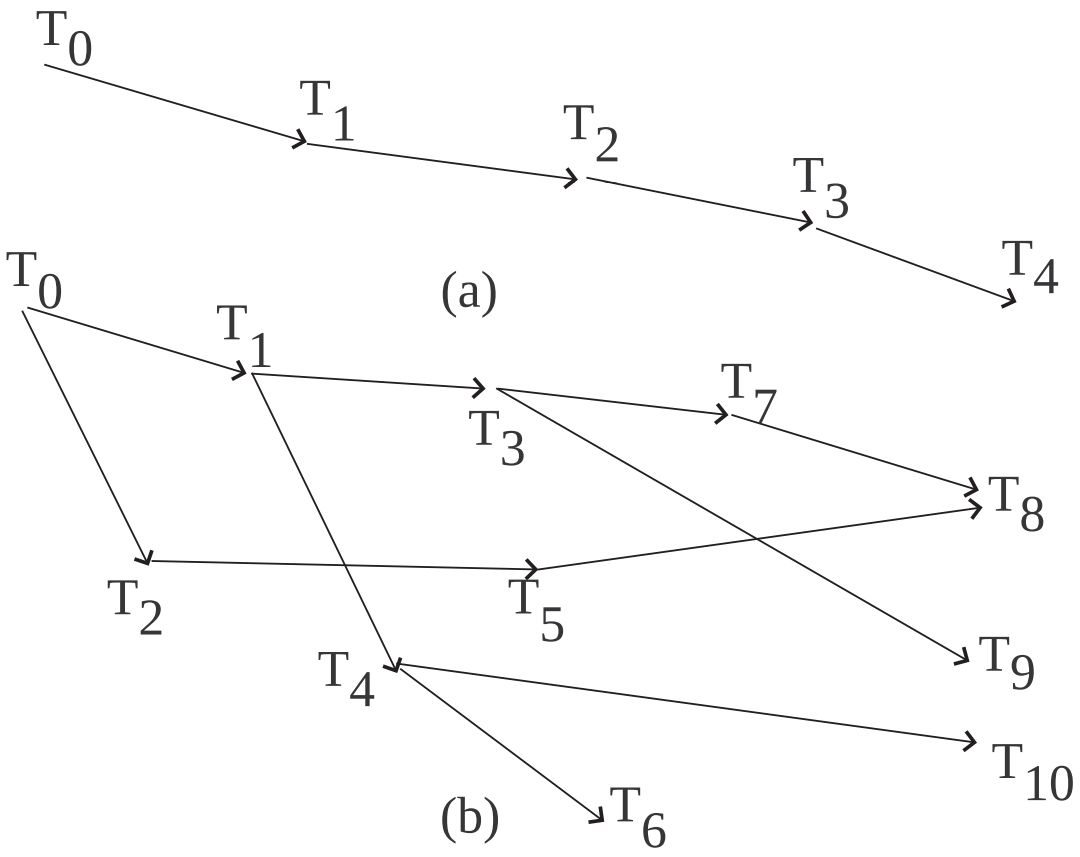
and refinement operation can propagate in many directions.

The *LEPP* graph can be sequentially processed with *O(n)* time complexity

In *2-D* meshes, parallel implementation can be handled in *O(logn)* complexity.

Longest Edge Propagation Path

(a) 2-D (b) 3-D



3-D Sequential Algorithm

Refinement of tetrahedron T in a *3-D* mesh

For each edge e of tetrahedron *T*
    Add *e* into *E_involved*
While *E_involved* is not empty
    Process an edge *e* from *E_involved*
    For each face *f* sharing edge *e*
        Select the longest-edge *LE* of face *f*
        If *LE* has not been processed before
            Add into *E_involved*
For each edge *e* in *E_involved* (*1-dimensiona*l)
    Bisect edge *e*
For each face *f* including any bisected edge *e* (*2-dimensional*)
    Partition face *f* according to bisected edges
For each tetrahedron including any partitioned face *f* (*3-dimensional*)
    Partition tetrahedron according to bisected faces

Algorithm for Distributed Environments

The parallel algorithm can be analyzed in three steps:

- Prepare propagation graph for refinement, (*DAG*) for *3-D* mesh .
- Find components which must be refined.
- Partition components according to the longest-edge bisection procedure.

The Algorithm(P processors, Tetrahedral Mesh *T*)

Distribute the Tetrahedral Mesh Structure among processors

Each processor Pi handles its local mesh structure

Process the Longest-Edge Algorithm locally

foreach edge *e* of tetrahedron *T* that needs to be refined
add edge e to the list of selected edges *E_selected*

LEPP algorithm

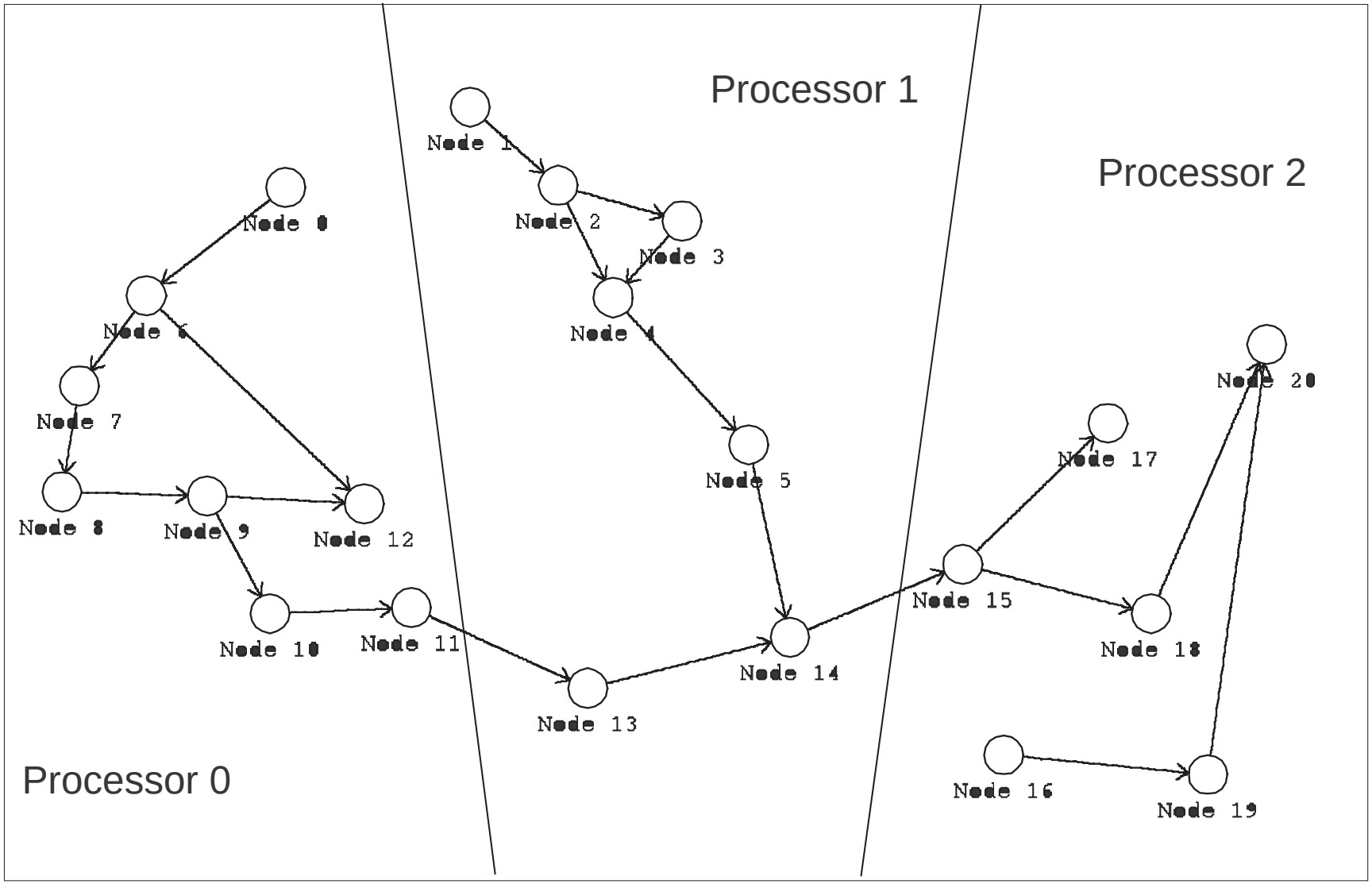
while all edge in *E_selected* are processed
if *LE* of the face *f* that edge *e* belongs is not selected
add edge *LE* to the list *E_selected*
add tetrahedron *T* that edge *e* belongs to the list of selected tetrahedra *T_selected*

Synchronize local Propagation Paths

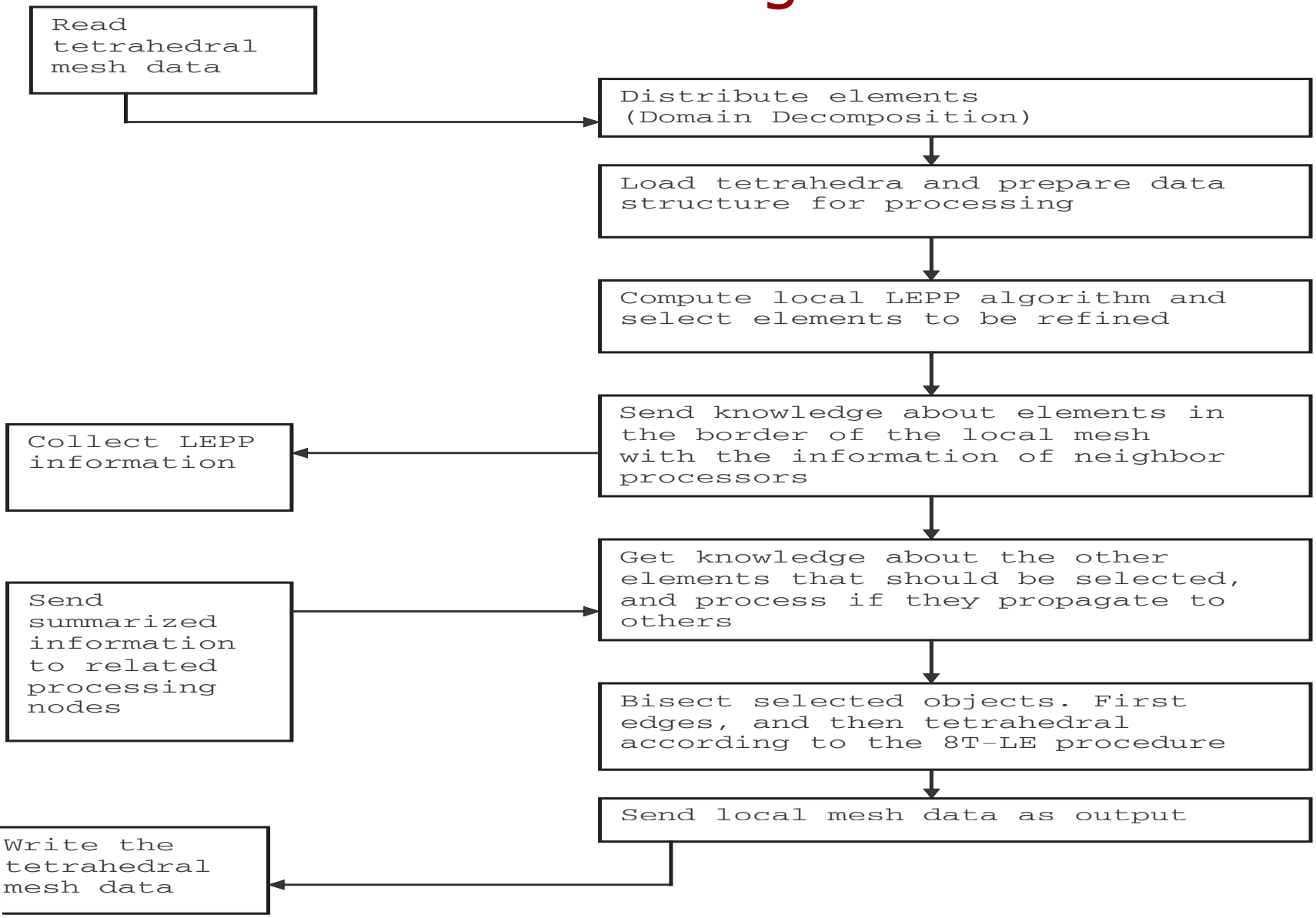
if local terminal point in the *LEPP* also belongs to another local mesh structure owned by processor *Pi*
inform processor *Pi* and
process the *LEPP* algorithm after synchronization

Bisect selected edges *E_selected*
Bisect selected tetrahedra *T_selected*
Collect local mesh structure from processing nodes *Pi*s

LEPP-Graph Partitioning and Synchronization



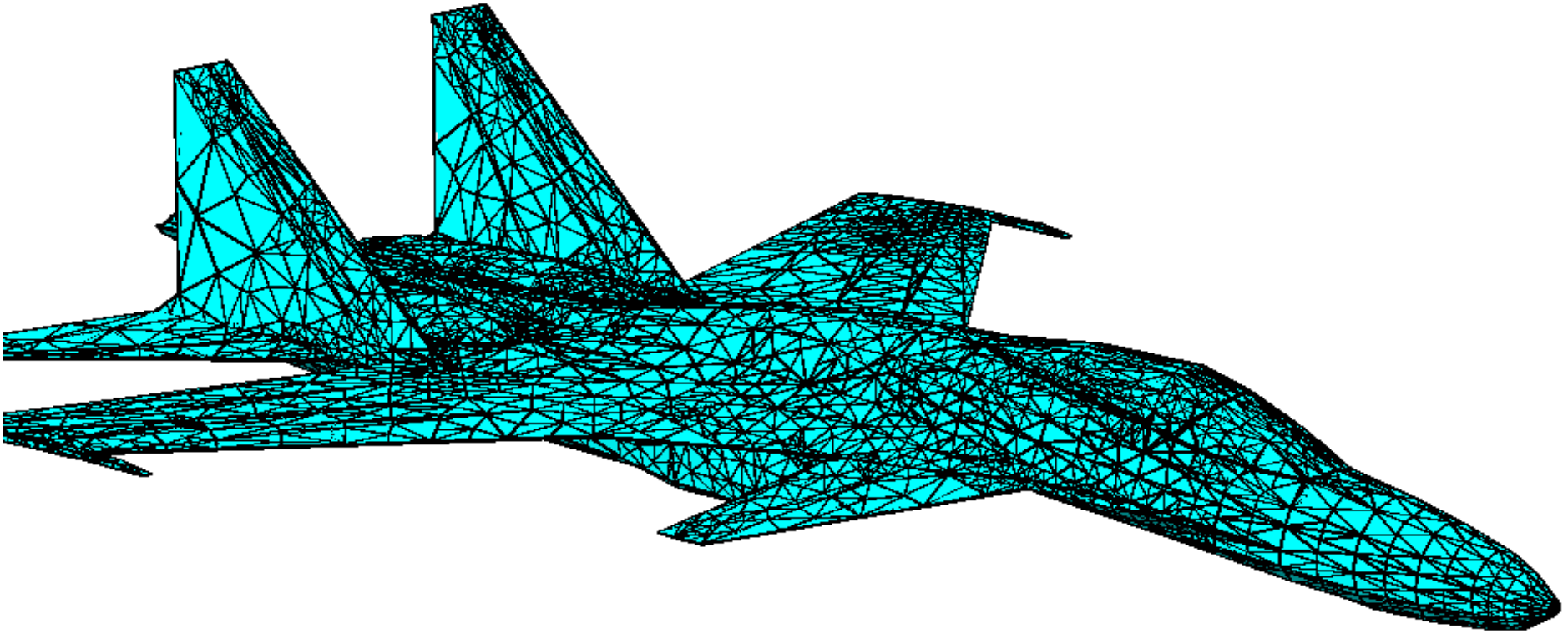
The PTMR Algorithm



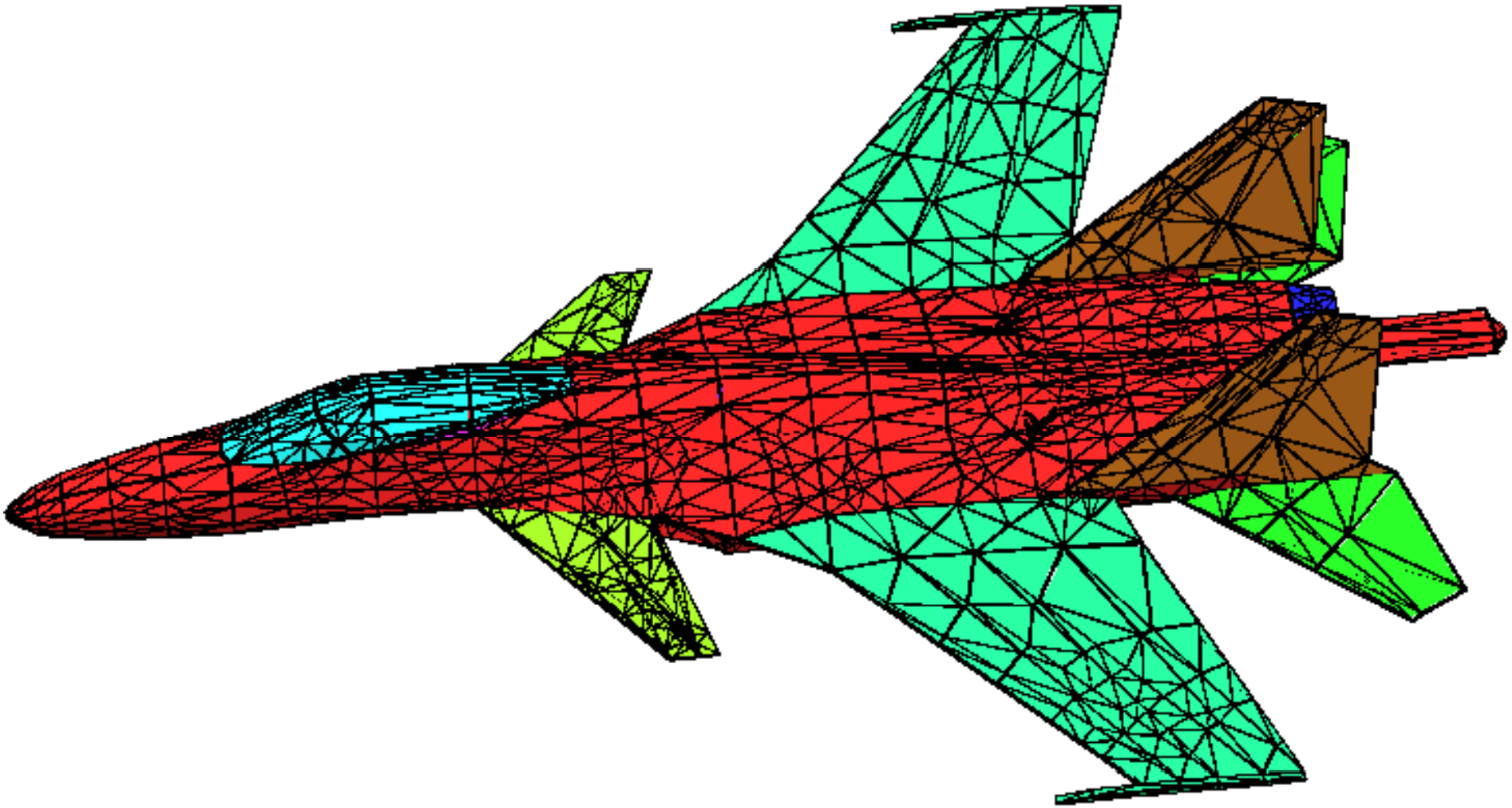
The PTMR Algorithm

- Partition and distribute tetrahedral mesh elements
- Compute initial tetrahedrons that should be refined
- Prepare the local *LEPP-Graph* sequentially and select the edges that should be refined
- Inform other processing nodes whether a border element in the local partition is selected
- Refine according to the *8T-LE* procedure.
- Collect mesh data with recently produced tetrahedrons

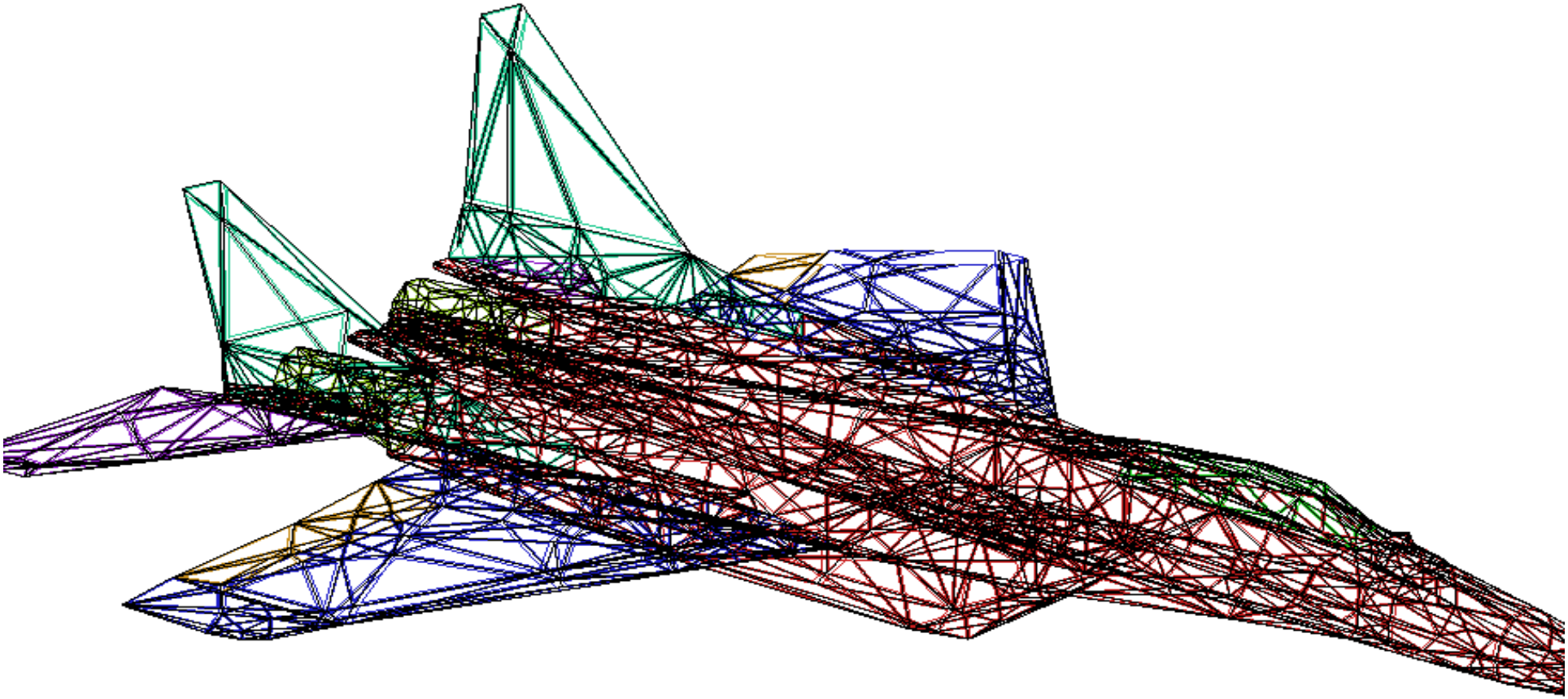
Example 70203 tetrahedra / 22568 vertices



Example 14904 tetrahedra / 4502 vertices

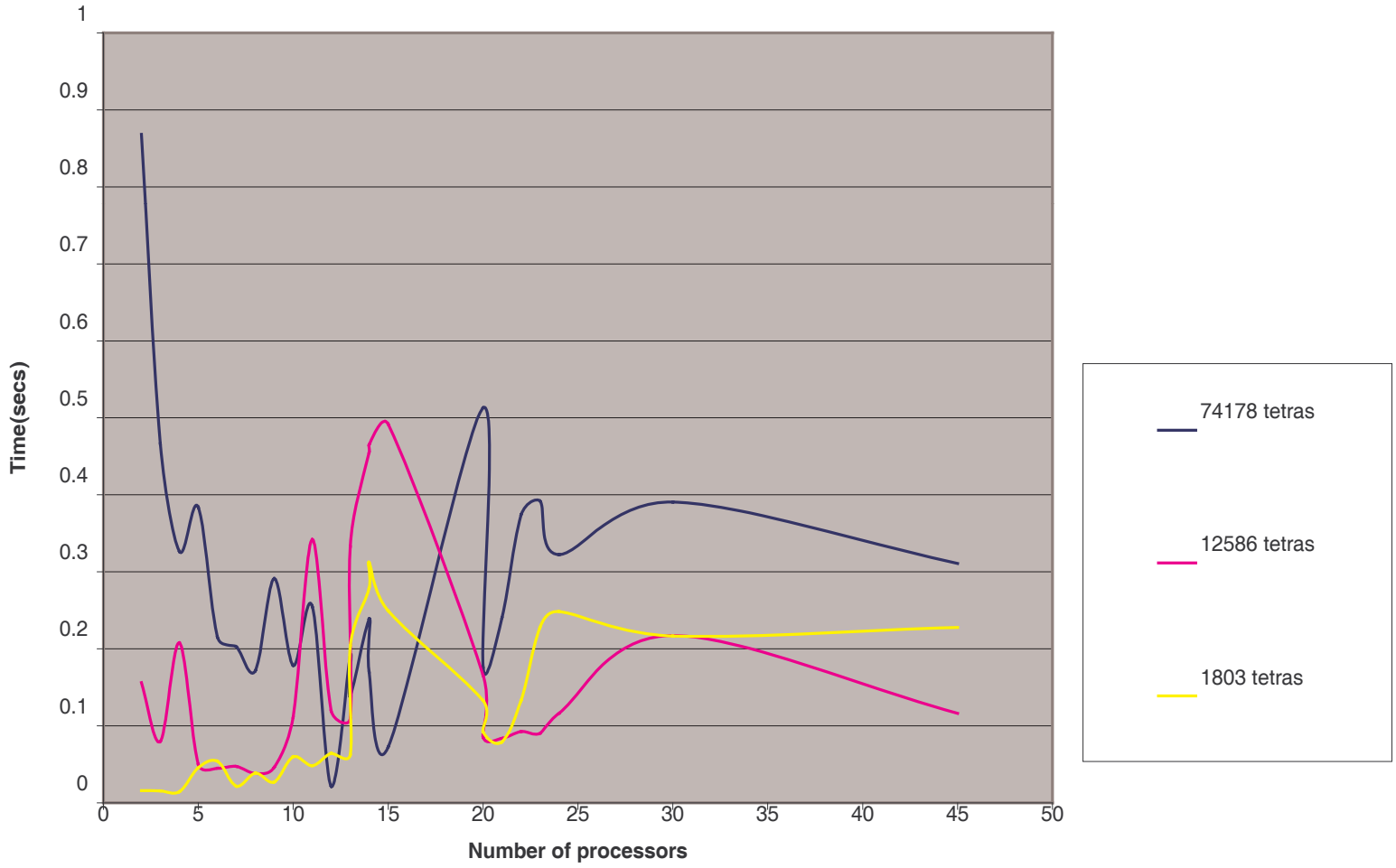


Example 1803 tetrahedra / 2021 vertices

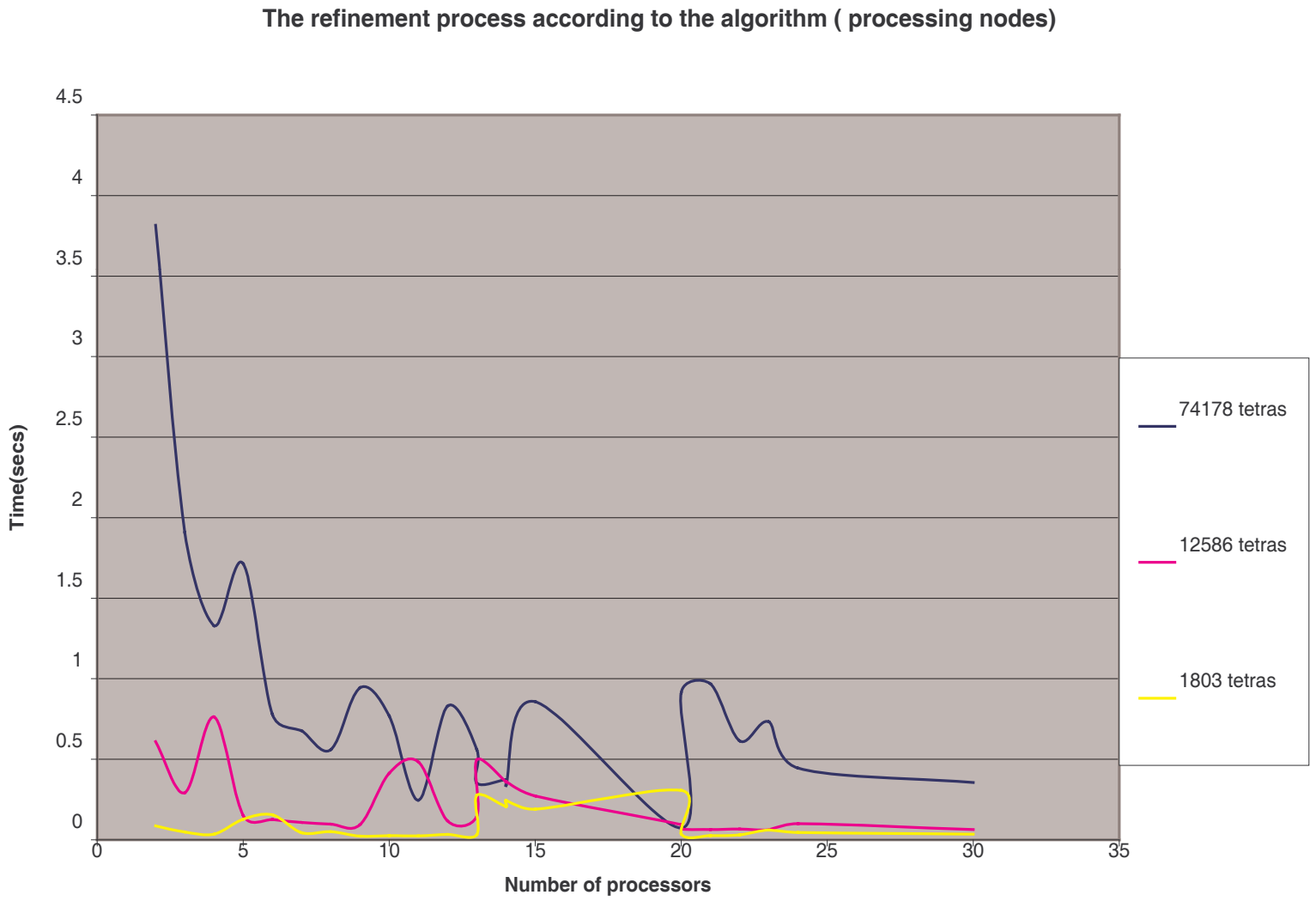


Empirical Results

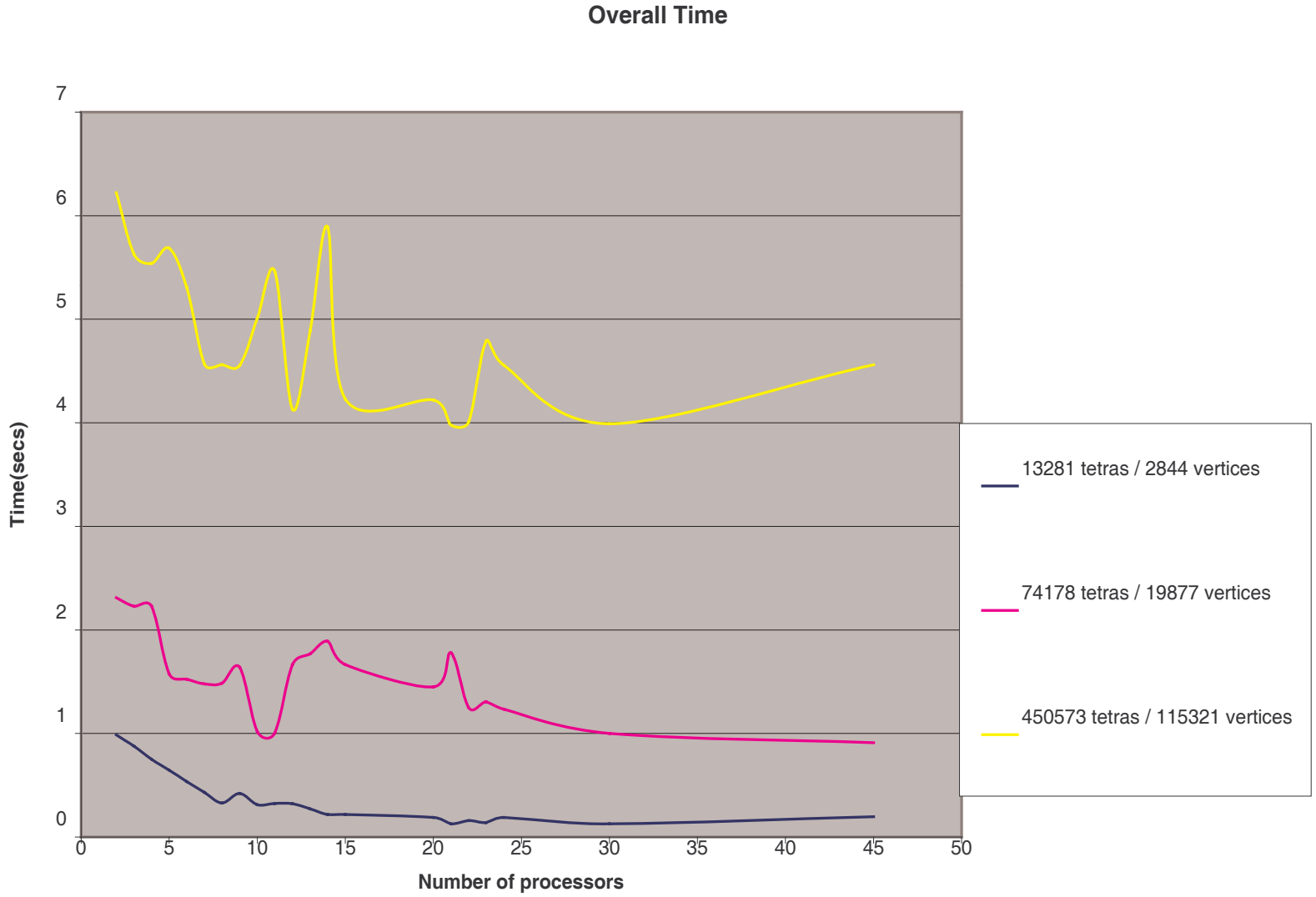
The refinement process according to the algorithm (elapsed time in the gateway node)



Empirical Results



Empirical Results



Conclusion & Contributions

A parallel mesh refinement algorithm for distributed environments is proposed.

Each processing node works over its local elements sequentially and synchronizes changes to update the overall mesh structure.

- A refinement framework for Finite Element problems
- The data structure for the mesh operations in distributed environments
- A brief study about the parallelization of *3-D* mesh refinement algorithms

This work will be submitted to

"The 8th International Workshop on High Performance Scientific and Engineering Computing (HPSEC-06)"