Graph grammar based unstructured mesh refinements

Maciej Paszyński

Anna Paszyńska, Krzysztof Podsiadło,


Department of Computer Science
AGH University of Science and Technology, Kraków, Poland

Oden Institute, The University of Texas at Austin
August 8, 2019

Outline

- Unstructured mesh refinements
- Rivara algorithm in 2D
- Graph grammar model
- Rivara algorithm in 3D
- Conclusions

Krzysztof Podsiadło, Albert Oliver Serra, Anna Paszyńska, Rafael Montenegro Armas, Maciej Paszyński, Graph-grammar mesh refinements and pollution simulations in Kraków area, to be submitted to **Engineering with Computers** (2019)

Motivation

Terrain mesh generator (implemented by Krzysztof Podsiadło)
https://github.com/Sambard/meshgen
The installation requires CMake in version 3.1 or higher and C compiler.
It generates the mesh with prescribed accuracy, using the Rivara algorithm
For a given set of coordinates it reads data from NASA database
-N <coordinate> North border of domain.
-S <coordinate> South border of domain.
-E <coordinate> East border of domain.
-W <coordinate> West border of domain.

Farr, T. G., Rosen, P. A., Caro, E., Crippen, R., Duren, R., Hensley, S., Kobrick, M., Paller, M., Rodriguez, E., Roth, L., Seal, D., Shaffer, S., Shimada, J., Umland, J., Werner, M., Oskin, M., Burbank, D., Alsdorf, D. The Shuttle Radar Topography Mission, **Reviews of Geophysics** 45, 2 (2005)

Terrain mesh generator parameters

-t <tolerance> Sets the refinement tolerance in meters. Defaut option is 5.

-s <requsted_size> Sets the lenght of initial triangles. It should be lower or equal than the shorter side of the domain.

-i <data_file> Indicates the input file in ASCII format.

-o <output_file> Base of filename in which results should be saved. The extension will be added depending on chosen output format.

-d <data_dir> Sets a location of directory with data files in SRTM format.

-p Write results into AVS UCP ASCII (.inp) format.

-m Write results into .smesh format.

-g Skip conversion into UTM coordinates.

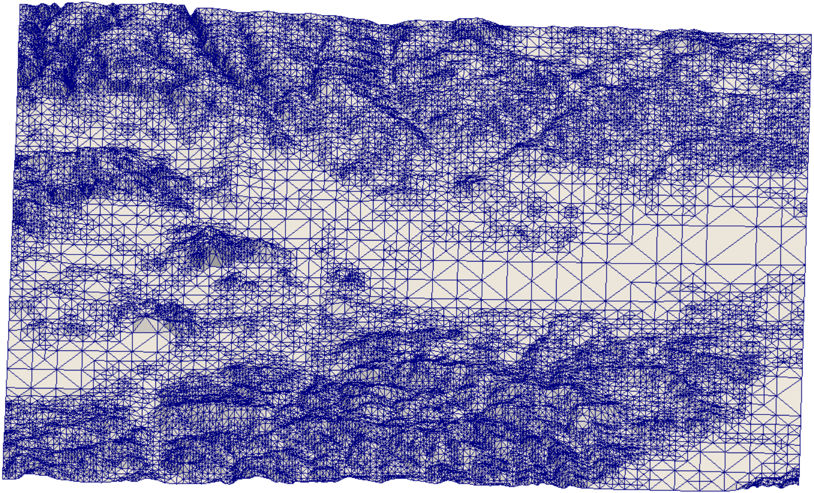
-u Convert to UTM coordinates before refinement

-U Convert to UTM coordinates after refinement (default).

-h Use 3D version of algorithm (2D is default).

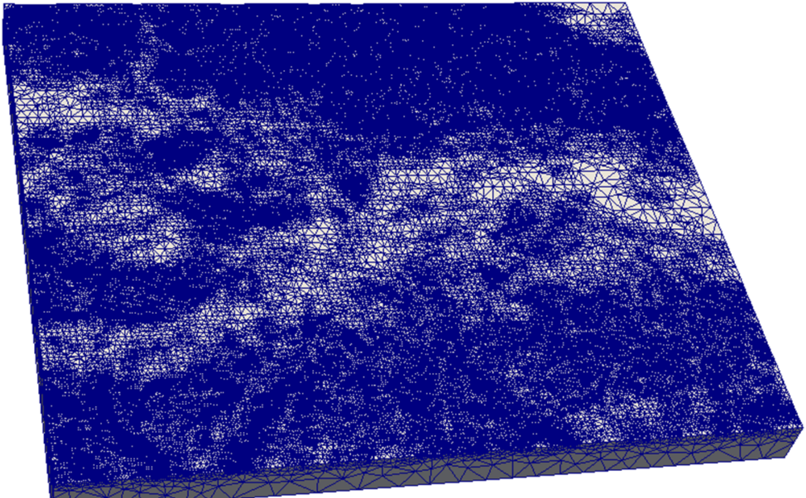
Unstructured mesh refinements

Unstructured mesh refinements



Unstructured mesh refinements

The obtained 2D mesh is then fit into the TETGEN that generates around 5 layers of three-dimensional tetrahedral elements, with a total of 491,843 tetrahedrals and 976 triangular elements on a "roof".



Pollution simulations

We focus on the advection-diffusion-reaction equations problem

$$\frac{\partial c_i}{\partial t} + u \cdot \nabla c_i - \nabla \cdot (K \nabla c_i) = s(c_i) \qquad (1)$$

where $c(x, t)$ is the unknown concentration of pollution in the area, namely the vector of four unknowns, representing the chemical components $c_1 = [SO2], c_2 = [SO4], c_3 = [NO4], c_4 = [NO3]$, $u(x, t) = [1, 0, 0]$ is a given wind velocity vector , reprsenting the west wind, $s(c)$ is chemical reactions part, where we assume linear model $s(c) = Ac$ where A is the chemical reactions matrix,

$$A = \begin{bmatrix} -0.15 & 0 & 0 & 0 \\ 0.15 & 0 & 0 & 0 \\ 0 & 0 & -0.3 & 0 \\ 0 & 0 & 0.3 & 0 \end{bmatrix} \qquad (2)$$

$$\frac{\partial c_i}{\partial t} + u \cdot \nabla c_i - \nabla \cdot (K \nabla c_i) = s(c_i) \tag{3}$$

Here K is the diagonal diffusion matrix

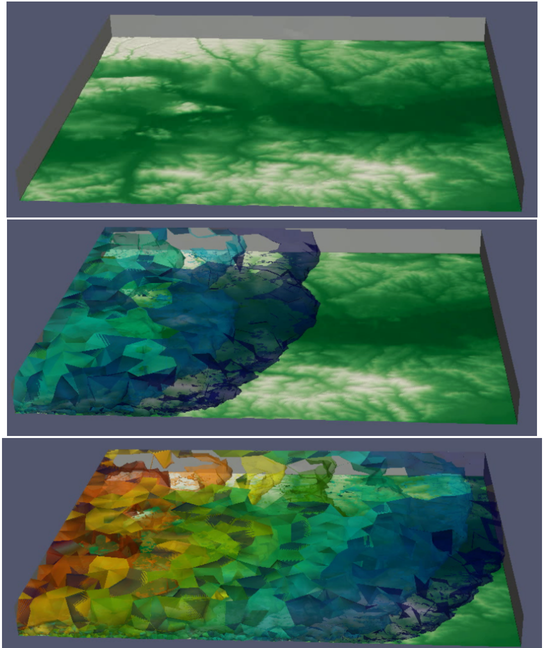
$$K = \begin{bmatrix} 8 * 10^{-6} & 0 & 0 \\ 0 & 8 * 10^{-6} & 0 \\ 0 & 0 & 4 * 10^{-6} \end{bmatrix} m^2/s \tag{4}$$

where the horizontal diffusion coefficient is eqaul to $8 * 10^{-6} m^2/s$, and the vertical diffusion coefficient $4 * 10^{-6} m^2/s$. The diffusion matrix is assumed to be identical for all the four species of the concentration field c. We assume that the pollutant comes from the western boundary, and it is blown inside the domain from the western boundary where we have the inflow of the western wind. Moreover, we have

$$n \cdot (K \nabla c) = -V^d c \tag{5}$$

at the terrain level, where $V^d = 1.3 * 10^{-3}$ m/s (so-called diagonal term of the deposition matrix).

Pollution simulations



Rivara algorithm 2D

M.-C. Rivara, New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations, **International Journal for Numerical Methods in Engineering**, 40 (1997) 3313-3324

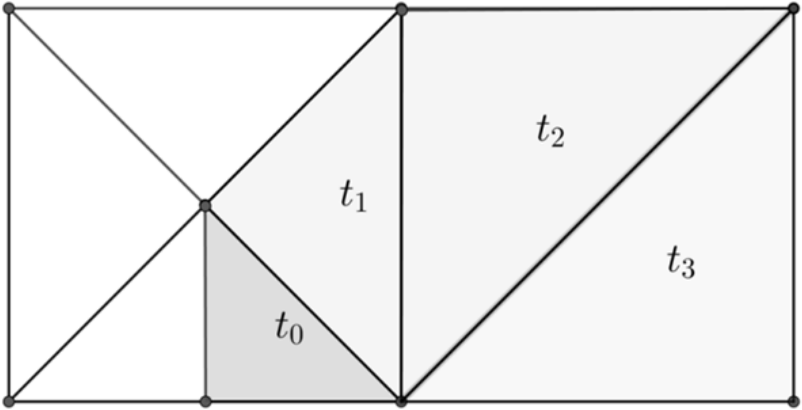
Rivara algorithm

```
function refine_mesh(T, tolerance)
T <- generate initial triangles
repeat
    for each triangle t in T
        if refinement condition in t is met
         (check_refinement(t, tolerance)==true) then
            T' <- refine triangle t
             (call refine(t))
            T <- T+T'
        endif
    endfor
until no refinements done
return T
```

Rivara algorithm

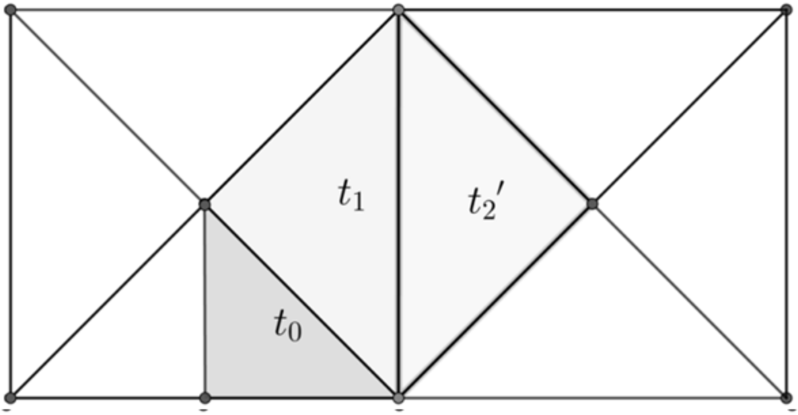
```
function refine (t)
t <- given triangle
t0 <- triangle lying by the longest edge of t
if t0 does not exists then
   Split t by its median
   return
endif
t00 <- triangle lying by the longest edge of t0
if t00 = t then
   Split t by its median
   Split t0 by its median
   return
end if
call refine(t0)
t0 <- triangle lying by the longest edge of t
Split t by its median
Split t0 by its median
```

Rivara algorithm



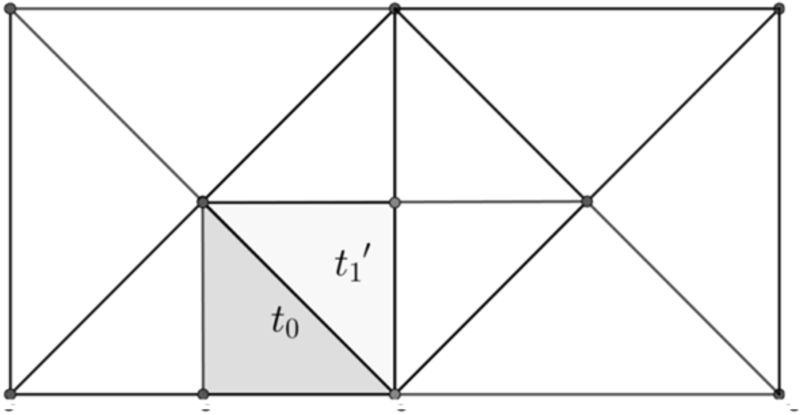
Intend to break t0 -> go to longest edge neighbor t1
-> go to longest edge neighbor t2 -> go to longest edge neighbor t3
-> the longest edge neighbor of t3 is t2 -> we will break both t2, t3

Rivara algorithm



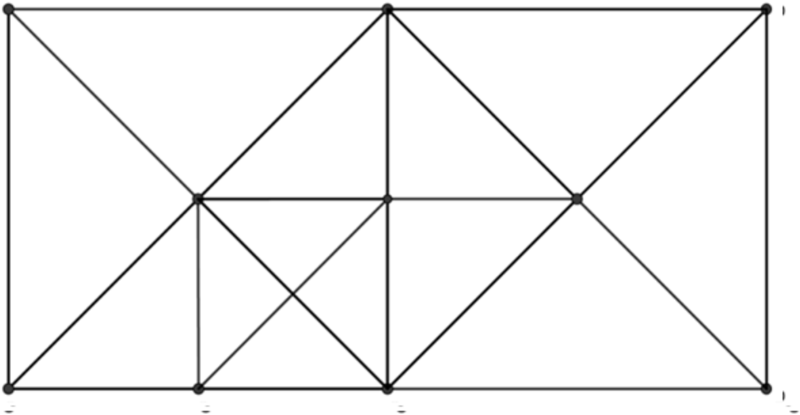
The longest edge neigbor of t2' is t1 -> we will break t1,t2'

Rivara algorithm



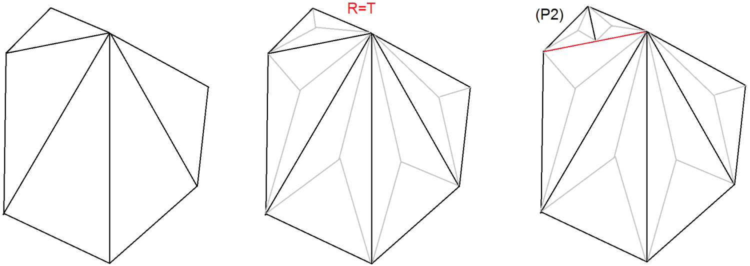
The longest edge neigbor of t1' is t0 -> we will break t0,t1'

Rivara algorithm



Graph grammar for 2D

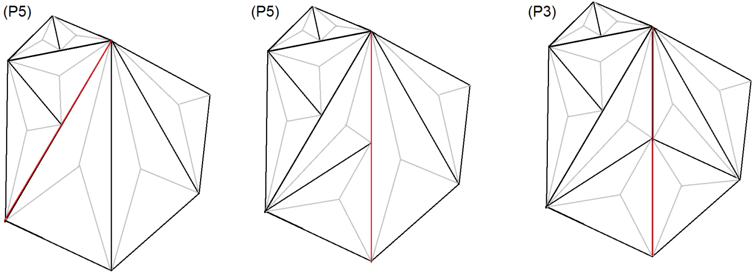
Graph grammar for mesh refinements



We want to refine top left corner element. We setup refinement flag R=T.

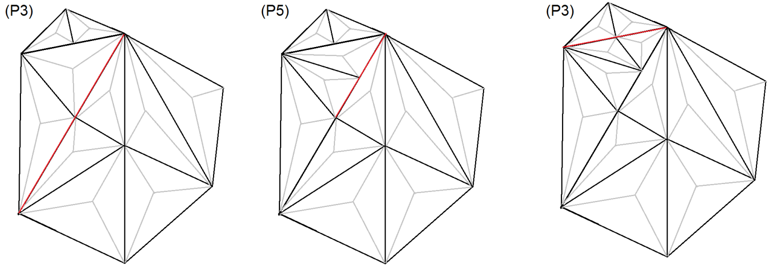
Production (P2) breaks the element towards the longest edge, the edge is labeled as having a hanging node.

Graph grammar for mesh refinements



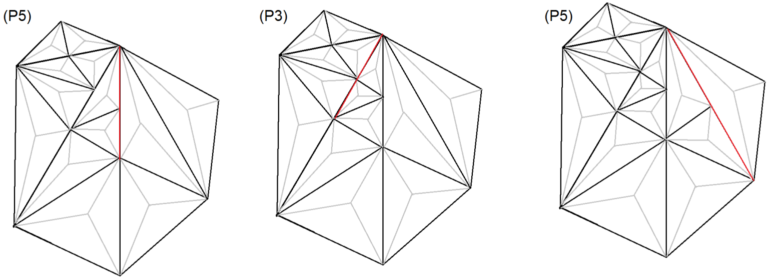
Production (P5) breaks the element with an edge having a hanging node, and it generates another edge with hanging node.
Production (P3) breaks the element with an edge having a hanging node, and it does not produce new hanging nodes.

Graph grammar for mesh refinements



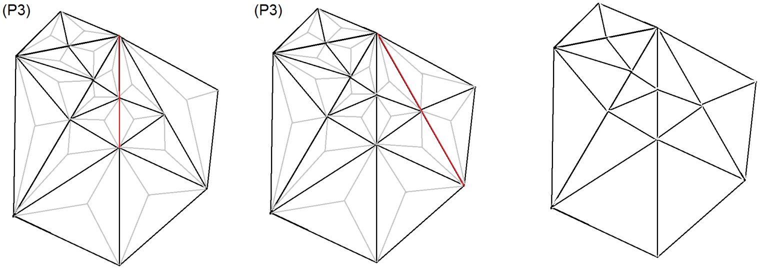
Production (P3) breaks the element with an edge having a hanging node, and it does not produce new hanging nodes.
Production (P5) breaks the element with an edge having a hanging node, and it generates another edge with hanging node.

Graph grammar for mesh refinements



Production (P5) breaks the element with an edge having a hanging node, and it generates another edge with hanging node.
Production (P3) breaks the element with an edge having a hanging node, and it does not produce new hanging nodes.

Graph grammar for mesh refinements



Production (P3) breaks the element with an edge having a hanging node, and it does not produce new hanging nodes.

Graph grammar for mesh refinements

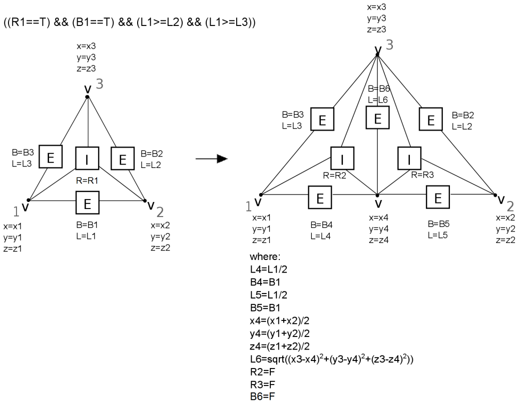


Figure: Production **(P1)** for the refinement of the element with the longest edge located on the boundary of the mesh.

Graph grammar for mesh refinements

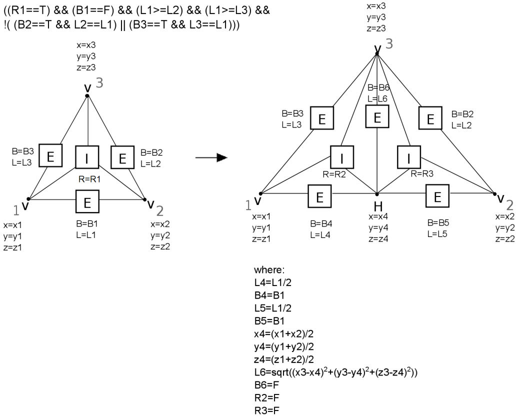


Figure: Production **(P2)** for the refinement of the element with longest edge located in the interior of the mesh.

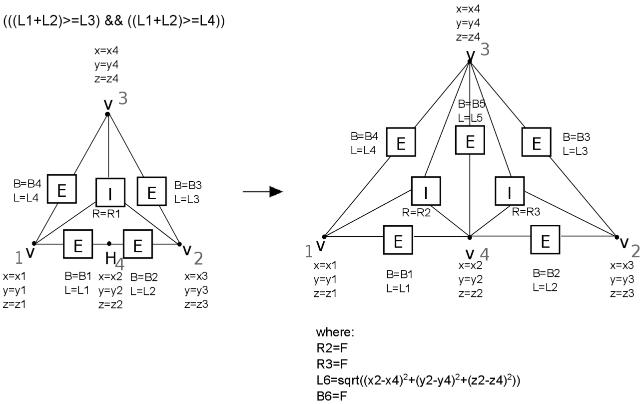
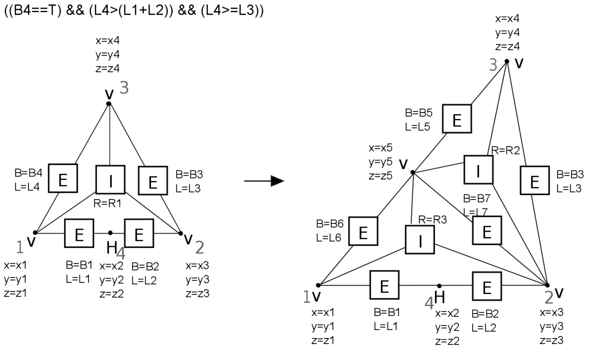


Figure: Production **(P3)** for the additional refinement of the element with the longest edge already broken with the hanging node, replacing the hanging node with the regular node.

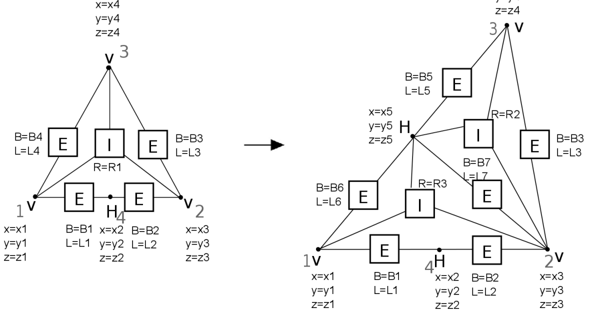


where:
L5=L4/2; B5=B4
L6=L4/2; B6=B4
x5=(x1+x4)/2
y5=(y1+y4)/2
z5=(z1+z4)/2
L7=sqrt((x3-x5)^2+(y3-y5)^2+(z3-z5)^2))
B7=F
R2=F
R3=F

Figure: Production **(P4)** for the additional refinements of the element with the longest edge located on the boundary, hopefully making the edge with the hanging node the longest edge.

Graph grammar for mesh refinements

((B4==F) && (L4>(L1+L2)) && (L4>=L3) &&
! ((B3==T) && (L3=L4)))



where:
L5=L4/2; B5=B4
L6=L4/2; B6=B4
x5=(x1+x4)/2
y5=(y1+y4)/2
z5=(z1+z4)/2
L7=sqrt((x3-x5)^2+(y3-y5)^2+(z3-z5)^2))
B7=F
R2=F
R3=F

Figure: Production **(P5)** for the additional breaking of an element with hanging node, when its longest edge is unbroken and located inside the mesh.

Graph grammar for mesh refinements

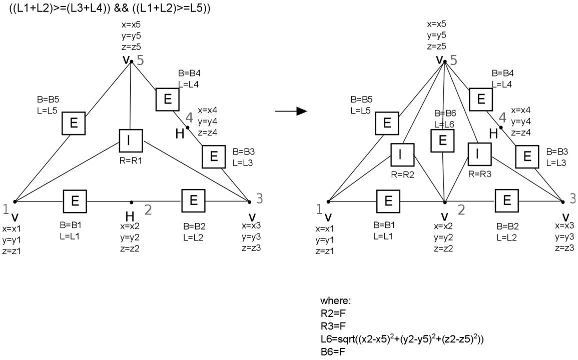


Figure: Production **(P6)** for an additional refinement of an element with two hanging nodes, intended to remove the hanging node located on the longest edge.

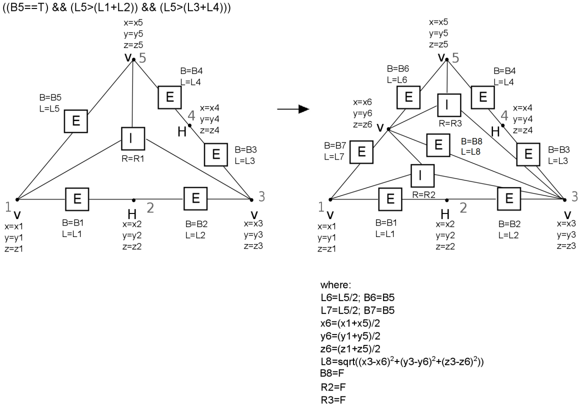


Figure: Production **(P7)** for an additional refinement of the element with two hanging nodes, and the unbroken longest edge located on the boundary.

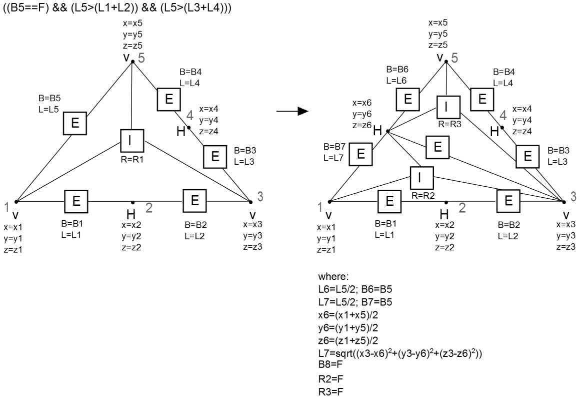


Figure: Production **(P8)** for an additional refinement of the element with two hanging nodes, and the unbroken longest edge located in the interior of the domain.

Conclusions

- Parallel implementation in GALOIS and / or GLUON and /or **GPU version** is needed
- Parallel mesh generator is promised as Krzysztof Podsiadło PhD thesis
- At this point we have 2D graph grammar model
- We will work on developing a 3D graph grammar model