Graph-grammar-based algorithm for asteroid tsunami simulations

Project duration: 01.03.2022–30.08.2022
Project manager: Dr Paweł Maczuga
Faculty of Computer Science, Electronics, and Telecommunications

AGH University of Krakow

RESEARCH UNIVERSITY EXCELLENCE INITIATIVE

INTRODUCTION:

We introduce a graph-grammar-based platform for performing tsunami simulations. There are the following novelties of our approach:

- **Mesh generation:** Using a novel graph-grammar-based approach, we adaptively generate the computational mesh covering the coastal area and the seabed.
- **Graph-grammar:** We propose only three graph transformations to express the longest-edge mesh refinement algorithm. Thus, our model is the generalization and simplification of the graph-grammar-based model of the Rivara algorithm which employed six graph-grammar productions for modeling of longest-edge refinements.
- **Simulation:** We employ the graph-grammar-based finite element method simulations of the tsunami propagation. We propose a simple stabilization procedure, and we verify it on a model hyperbolic problem.

METHODS AND RESULTS:

Longest-edge refinement algorithm

Mesh refinement is the process of subdividing an element of a mesh to obtain a finer mesh. During the mesh refinement process, the original nodes are not removed, and the topology of the original mesh is preserved. Usually, we refine the elements where the error is large in order to compute a better numerical solution.
Longest-edge refinement algorithm always breaks the element along its longest edge, generates a new point in the middle of the longest edge and generates the two new elements elements. Algorithm ensures that the new elements are the highest quality and ensures the conformity of the mesh (no presence of the hanging nodes).
In this work, we propose the graph representation of the computational mesh, where the triangle's interior is connected with edges (not with nodes, like in our previous research). This allows to reduce the number of graph-grammar production to 3 (instead of 6 in previous works), which in turn allows to implement faster algorithms.
The productions are:
P0: Sets refinement flag for elements to be broken (we do not illustrate this simple production with a figure)
P1: Bisects the triangle by an edge that is not broken

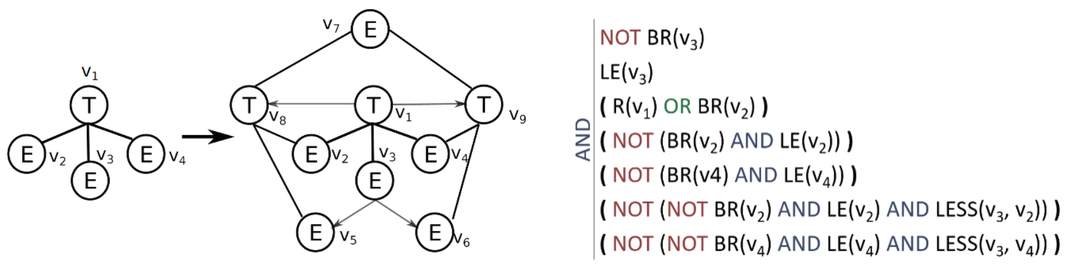


Figure 1. Production P1 with predicates of applicability.

P2: Bisects the triangle by a broken edge

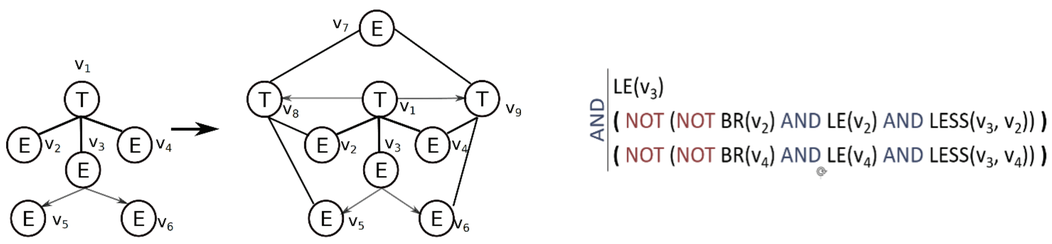
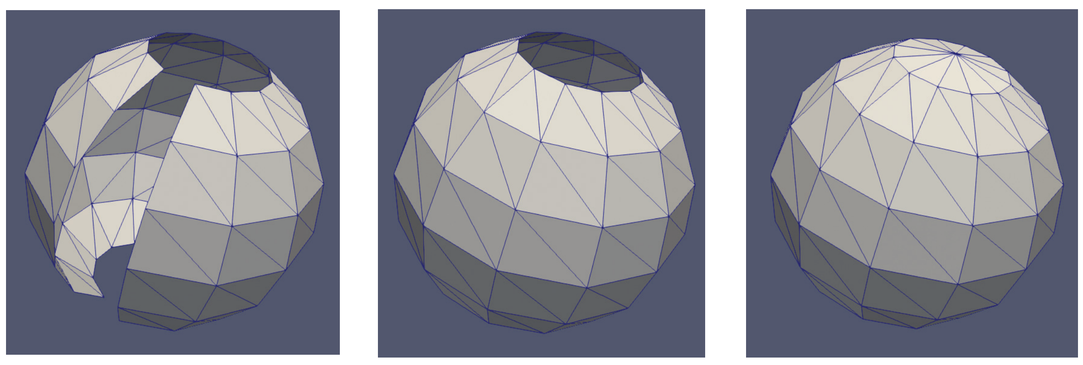


Figure 2. Production P2 with predicates of applicability.

Where:
LE – longest edge
R – refinement flag
BR – on boundary
LESS(v1, v2) – v1 is shorter than v2

Generation of a sphere mesh

Our goal is to generate the mesh of a sphere representing the whole Earth as a base for future simulations. We do that by firstly creating initial, low resolution mesh of a whole sphere and then we adapt it using longest-edge refinement algorithm. The initial mesh is already a complete sphere including poles. It is represented as a hypergraph, as described before, but each vertex has two types of coordinates – cartesian (xyz) and geographic (uv).
To generate the initial mesh, we start from rectangle mesh without poles and one horizontal segment, where vertices are equally evenly spaced.
In resulting mesh, the leftmost vertices with u=-180 are in fact the same as the missing 180 vertices on the right. Therefore, we fill in the missing segment by connecting corresponding vertices in order to have proper poles-less sphere in 3D. Lastly, we add two new vertices for poles in coordinates along with appropriate triangles.



Longest-edge refinement of a sphere mesh

Ultimately, we use the longest-edge refinement algorithm on generated mesh to create high-resolution one that may be used in the simulation. The algorithm operates on a single triangles, and so the only issue is to specify:

- How the distance between two vertices is calculated. The longest edge according to this distance will be broken.
- How and where to add new vertex after breaking an edge.

Tsunami simulations with the shallow water equations

Tsunami is modelled using the following non-linear wave equation:

$$\frac{\partial^2 u}{\partial t^2} - \nabla(\mathrm{g}(\mathrm{u}-\mathrm{z})\nabla u) = 0$$

u – water level z – seabed g – gravitational acceleration
The equation is solved by using explicit time integration scheme.

To verify the simulation, we have prepared simple environment: square pool filled with water, according to $2e^{(-r^2)}+2$. Then, we performed a tsunami simulation on the Baltic Sea, where the mesh was generated using previously described longest-edge refinement algorithm.

CONCLUSION:

We proposed a graph-grammar-based platform for performing simulations of asteroid tsunamis. We introduced the graph representation of the computational mesh, where the triangle's interior is connected with edges (not with nodes, like in our previous research). This representation allowed to reduce the computational cost of finding the left-hand-side of the production. We employed the longest-edge refinement algorithm implemented by three graph-grammar productions. The framework was implemented in Julia. For discretization, we employed the finite element method with triangular elements of the generated mesh. We use linear basis functions to discretize the problem in every time step of the explicit simulation. It employed LU factorization of the mass matrix and forward/backward substitutions for the consecutive time steps. To verify our platform, we performed graph-grammar-based simulations of the tsunami caused by the asteroid falling into the Baltic sea. This simulation takes around 1 hour on a laptop.
Future work may involve parallelization into shared memory and hybrid memory Linux clusters. Additionally, we plan to employ inverse algorithms to solve several related inverse problems. In our future work, we also plan to generalize the graph-grammar model into 3D.

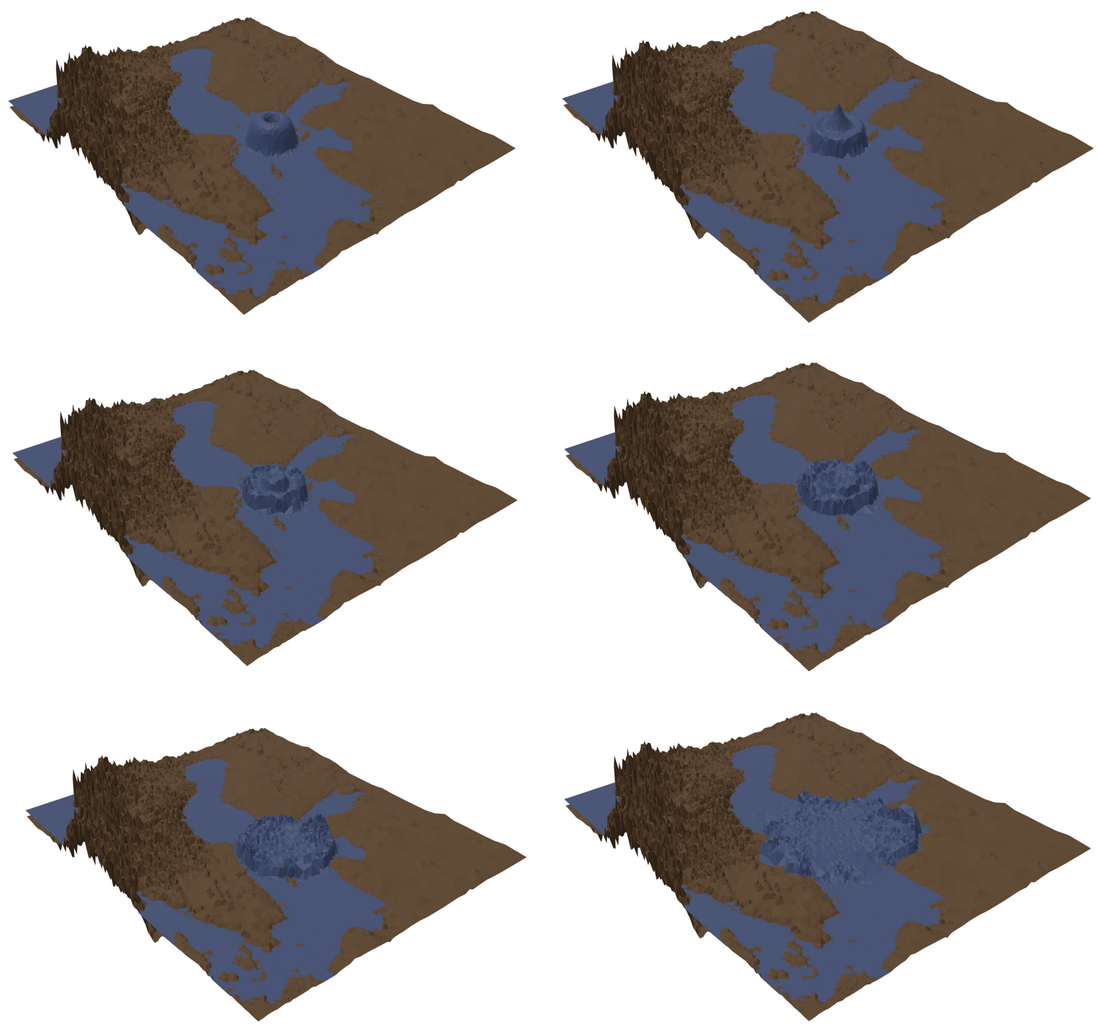


Figure 3. Tsunami on Baltic Sea