Graph grammar model for mesh refinements for high-order polygonal discontinuous Petrov-Galerkin method

Paweł Maczuga[1][0000−0002−5111−6981], Mateusz Dobija[2][0000−0003−4557−3534], Kamila Ćwikła[1], Szymon Mamoń[1], Albert Oliver-Serra[3][0000−0002−3783−8670], and Anna Paszyńska[4][0000−0002−0716−0619]

[1] Faculty of Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, Kraków, Poland
pawel.maczuga@agh.edu.pl
[2] Faculty of Metals Engineering and Industrial Computer Science, AGH University of Krakow, Al. A. Mickiewicza 30, 30-059, Krakow, Poland
mateusz.dobija@agh.edu.pl
[3] University of Las Palmas de Gran Canaria (ULPGC), Gran Canaria, Spain
albert.oliver@ulpgc.es
[4] Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, ul. prof. Stanisława Łojasiewicza 11, Kraków, Poland
anna.paszynska@uj.edu.pl

Abstract. One of the commonly used methods for performing simulations is the high-order polygonal discontinuous Petrov-Galerkin method. In the paper, the graph grammar approach for modeling mesh refinements for the polygonal discontinuous Petrov-Galerkin method is introduced. The computational mesh is represented by hypergraphs, and mesh adaptations are represented by graph grammar productions.

Keywords: high-order polygonal discontinuous Petrov-Galerkin method · mesh adaptations · graph grammar.

1 The high-order polygonal discontinuous Petrov-Galerkin method

One of the commonly used finite element methods (FEM) [1], [2] is the high-order polygonal discontinuous Petrov-Galerkin method (PolyDPG) [3], which can solve a variety of linear problems. PolyDPG methods are supported by a strong theoretical foundation that guarantees stability and high-order convergence. They can also handle discontinuous material properties while still maintaining stability. In FEM, in order to increase the accuracy of the obtained solution, a refinement of the mesh may be needed. The high-order polygonal discontinuous Petrov-Galerkin methods provide an arbitrary-order a posteriori error estimator, making adaptive mesh refinement easy to implement. This advantage is especially valuable for polygonal elements, as it eliminates the need

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

for special techniques to manage hanging nodes, making the method well suited for applications such as topology optimization and dynamic fracture [3]. In this paper, we will focus on the polygonal adaptive strategy used in PolyDPG.
The polygonal adaptive strategy can be described as follows:

- if a polygonal element fulfills the so-called refinement criterion and should be refined, it is broken into quadrilaterals by using the centroid of polygonal elements and polygon edge midpoints,
- if a "big" neighbor element needs to be refined, it is split into quadrilaterals assuming that all adjacent collinear edges constitute a single "big" edge.

Figure 1 presents exemplary mesh refinements with a polygonal adaptive strategy. The red dot indicates that the element needs to be refined. For comparison, Figure 2 presents an exemplary traditional quadtree mesh with constrained nodes and quadrilateral mesh generated using the polygonal adaptive strategy, respectively.

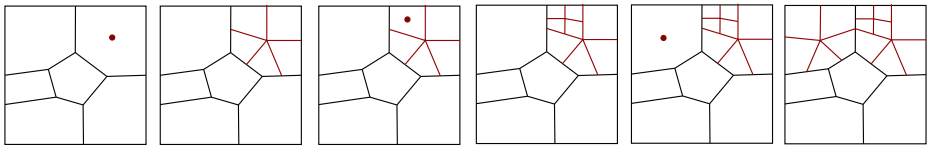

Fig. 1. Refinements with polygonal adaptive strategy

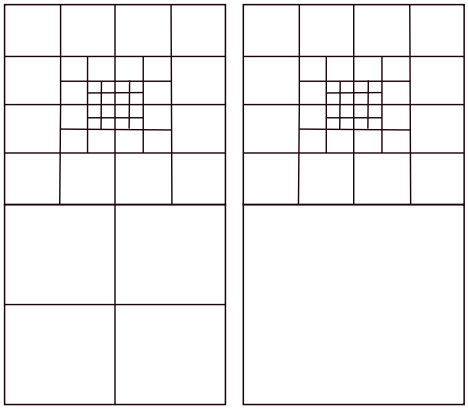

Fig. 2. Traditional quadtree mesh with constrained nodes and quadrilateral mesh generated using the polygonal adaptive strategy

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

2 Graph grammars

Graphs are commonly used to represent complex structures, and graph grammars can describe the changes in these structures. Graph grammar consists of an initial graph and a set of so-called graph grammar productions, which allows to "rewrite" the graph, together with embedding definition. The graph grammar production consists of the left-hand-side graph and the right-hand-side graph and should include a definition of embedding. The production can also have an applicability predicate: additional conditions (related to attribute values, not to the graph structure) specifying when production can be applied. In order to rewrite a graph H, we have to select a production from the set of graph grammar productions $(\alpha \rightarrow \beta)$, where α and β are graphs, replace an occurrence of α in H by β and then embed β into the rest of H, according to predefined embedding, that is, according to the established edges between the nodes of β and the remaining nodes of H.
There are many types of graph grammars - the main difference between them lies in the type of graph it operates and on the type of embedding transformation [9]. The easiest to implement with great power of generation are graph grammars with invariant embedding, like for example hypergraph grammar [7], [8] and composition graph grammar [5], [6].
In our approach, we will use hypergraph to represent computational mesh and hypergraph grammar to represent mesh transformations.

2.1 Hypergraph grammars

Hypergraphs are defined as a set of nodes and a set of hyperedges. A hyperedge can connect an arbitrary number of nodes. Nodes and hyperedges can be labeled and have attributes (variables connected with graph nodes and hyperedges). Figure 3 (left panel) presents an exemplary hypergraph with four nodes labeled v with attributes x, y, z, one hyperedge labeled Q with attribute R, and four hyperedges labeled E with attributes R and B.
The hypergraph grammar production consists of the left-hand-side hypergraph and right-hand-side hypergraph. In order to define invariant embedding, some of the left-hand-side graph nodes are distinguished by unique numbers (so called external nodes). Each external node from the left-hand-side-graph should have corresponding node (with the same number) in the right hand-side-graph. An exemplary production is presented in figure 3. The application of a production $(\alpha \rightarrow \beta)$ to a hypergraph H consists of replacing a subhypergraph of H isomorphic with α by a hypergraph β and replacing external nodes of the removed subhypergraph isomorphic with α by the corresponding external nodes of β. Figure 4 shows an exemplary graph H and graph H after applying the production of Figure 3.

2.2 Graph grammars and their application to mesh generation

The first attempt to model mesh refinements using graph grammar was made in 1997 in [4]. In the paper, quasi-sensitive graph grammars were used to model uni-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

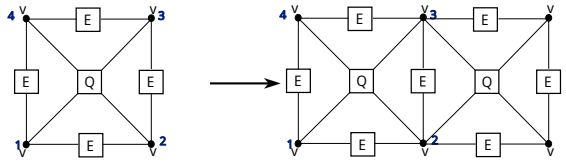

Fig. 3. Exemplary production.

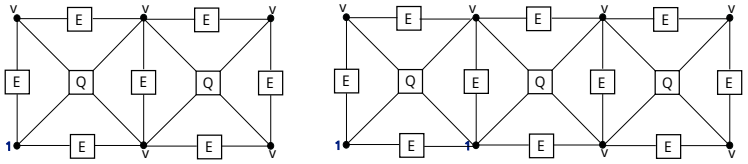

Fig. 4. Initial graph and graph after applying production from fig. 3.

form refinement of triangular meshes. In [10] composition graph grammars were used to model mesh generation using the h-adaptive finite element method with rectangular meshes. In [11] composition graph grammars were used to model mesh generation in the h-adaptive finite element method with mixed rectangular and triangular element meshes. In [12] the hypergraph grammar was used to model mesh generation in hp-adaptive finite element method with rectangular elements. Further works introduced graph grammars modelling mesh adaptation for triangular meshes with the longest-edge refinement algorithm [13], as well as meshes with polygonal elements [14]. All these models concern uniform mesh adaptation for triangular meshes, traditional mesh adaptation (with hanging nodes) with different types of polygonal elements, and mesh adaptation for triangular meshes with the longest-edge refinement algorithm. In this paper, we introduce the hypergraph grammar for modeling mesh adaptation using the polygonal adaptive strategy. Using graph grammars is motivated by the need to formally, modularly, and transparently model evolving mesh topologies. For the mesh refinement, the graph grammar approach offers a natural abstraction of local refinement operations, explicit and verifiable topology handling, support for adaptivity and hierarchy, as well as strong foundations for parallel and multi-level methods.

3 Hypergraph grammar for polygonal adaptive strategy

In our approach, polygon elements will be represented by hypergraphs. Hypergraphs representing quadrilateral, pentagonal and hexagonal elements are presented in figures 3 and 6. The hyperedges with label **Q**, **P** and **S** represent the interior nodes of quadrilateral, pentagonal, and hexagonal elements. The attribute **R** of hyperedges with labels **Q**, **P** and **S** is a boolean value defining if an element needs to be broken ($R = 1$ if an element is marked for refinement, $R = 0$ in the other case). The hyperedges with label **E** represent the edges of the

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

polygons. Its attribute **B** is a boolean value defining if the edge is a boundary edge ($B = 1$ for a boundary edge, 0 for a shared one) and attribute **R** is a boolean value defining if an edge needs to be broken ($R = 1$ if an edge is marked for refinement, $R = 0$ in the other case).

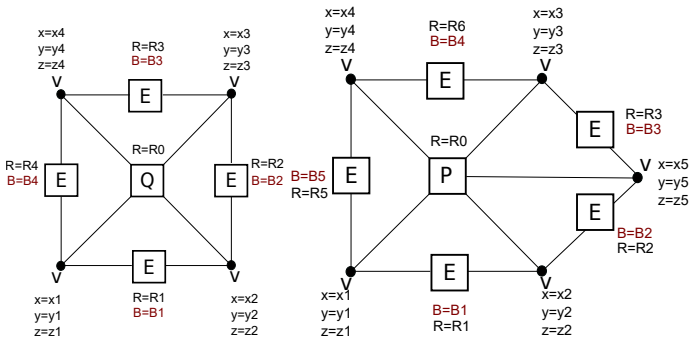

Fig. 5. Hypergraphs representing quadrilateral and pentagonal elements.

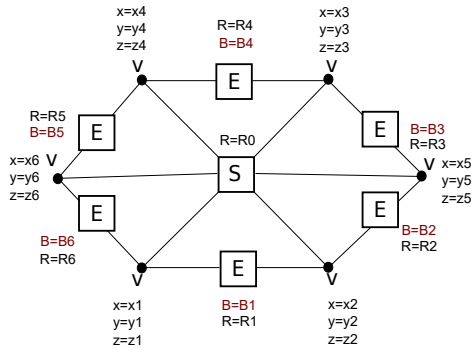

Fig. 6. Hypergraph representing hexagonal element.

The process of refinement of polygonal mesh can be modeled as the four step process:

- Mark an element for breaking
- for marked element mark its edges for breaking
- perform breaking of the marked edges
- break the element marked for refinement, with all edges broken

Production **P0** marks a **quadrilateral element** for refinement if it meets the refinement criterion (**RFC** = **1**) - it sets the value of the attribute **R** of the hyperedge with the label **Q** to **1** (fig. 7).

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

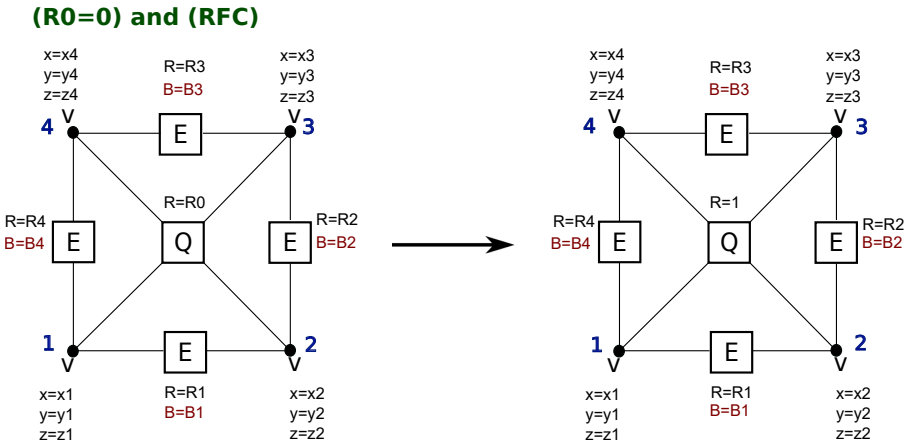

Fig. 7. Production **P0** - marking quadrilateral element for breaking.

Production **P1**, presented in figure 8 marks the edges of the **quadrilateral element**, which is marked for refinement ($\mathbf{R0} = \mathbf{1}$), for breaking - it sets the value of attribute **R** of each hyperedge with the label **E** to **1**.

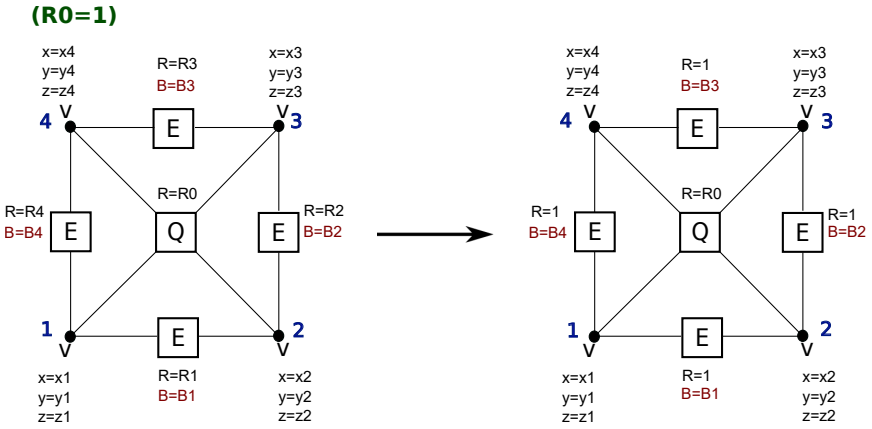

Fig. 8. Production **P1** - marking edges of quadrilateral element for breaking.

Production **P2**, shown in fig 9 breaks the shared edge marked for refinement if the edge was already broken by the neighboring element - there exists a "big" edge between nodes 1 and 2 and two small edges. The nodes $1, 2, 3$ can be recognized by coordinates - the coordinates of node 3 are a mean value of the coordinates of nodes 1 and 2. In this case, production removes the "big" edge from the element - it removes the hyperedge labeled by **E** between nodes 1 and

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

2. Production sets the value of the attribute **R** of each remaining hyperedge with the label **E** to **0**.

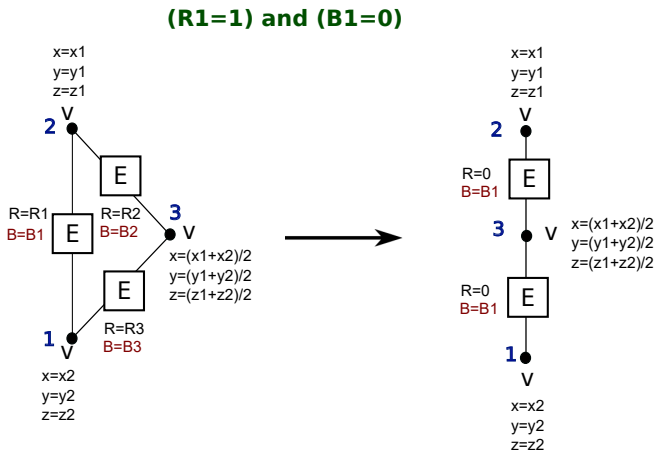

Fig. 9. Production **P2** - breaking the shared edge marked for refinement, if the edge was already broken by the neighboring element.

Production **P3** breaks the shared edge marked for refinement, if the edge was not broken by the neighboring element - there exists only a "big" edge between nodes 1 and 2. In this case, production adds two new "small edges" - it adds a new node (with coordinates calculated as mean value of coordinates of existing nodes 1 and 2) and adds two hyperedges with label E - one hyperedge between node 1 and the new node and the second hyperedge between node 2 and the new node. Production sets the value of the attribute **R** of all hyperedges with the label **E** to **0**. Production $P3$ is presented in figure 10.

Production **P4** breaks the boundary edge ($B1 = 1$) marked for refinement ($R1 = 1$). In this case, production removes "big" edge between nodes 1 and 2 and adds two new "small edges" - it adds a new node (with coordinates calculated as the mean value of coordinates of existing nodes 1 and 2) and adds two hyperedges with label E - one hyperedge between node 1 and the new node and the second hyperedge between node 2 and the new node. Production sets the value of the attribute **R** of all new hyperedges with label **E** to **0**. Production $P4$ is presented in figure 11.

Production **P5** breaks the quadrilateral element marked for refinement if all its edges are broken. Production removes the hyperedge **Q** representing the "big" element, and creates a new node in the center of the quadrilateral element. Production creates four new hyperedges with the label Q, representing the interiors of the newly created elements, and sets the values of the attribute **R** of the newly created hyperedges with the label **Q** to **0**. It also creates four new hyperedges with label **E** with $\mathbf{R} = \mathbf{0}$ and $\mathbf{B} = \mathbf{0}$. Production $P5$ is presented in

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

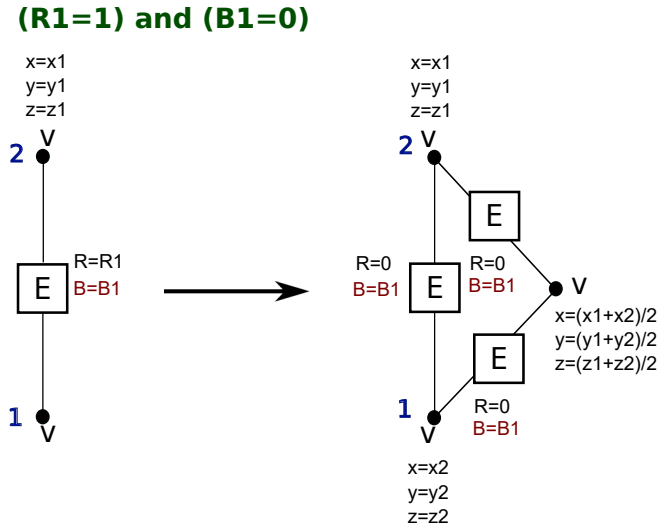

Fig. 10. Production **P3** - breaking the shared edge marked for refinement, if the edge was not broken by the neighboring element.

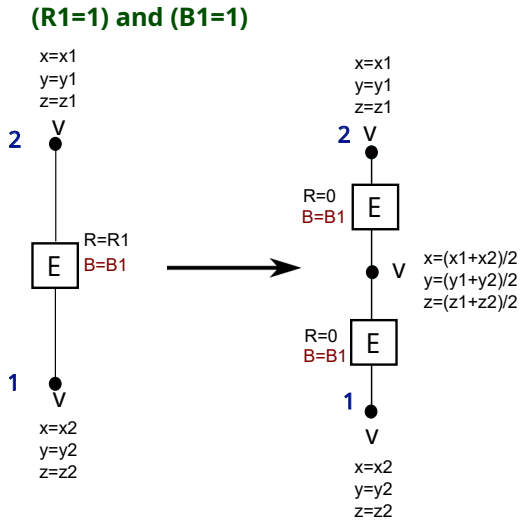

Fig. 11. Production **P4** - breaking the boundary edge marked for refinement.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

figure 12.

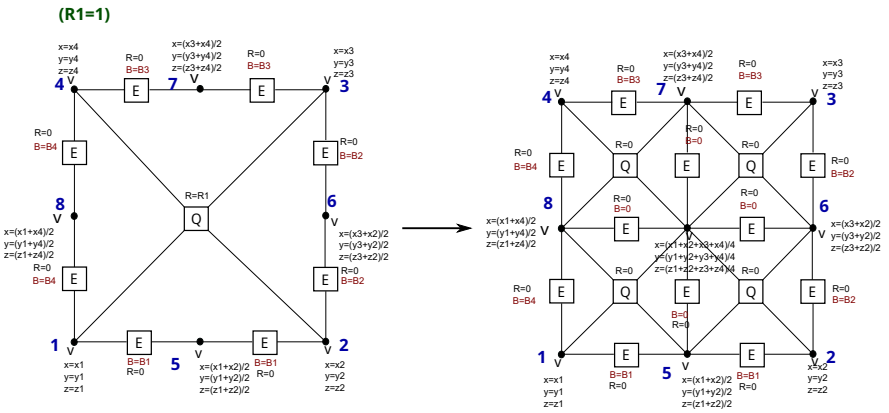

Fig. 12. Production **P5** breaking the quadrilateral element marked for refinement

Production **P6**, presented in fig. 13, marks a **pentagonal element** for refinement if it fulfills the refinement criterion - it sets the value of the attribute **R** of the hyperedge with the label **P** to **1**

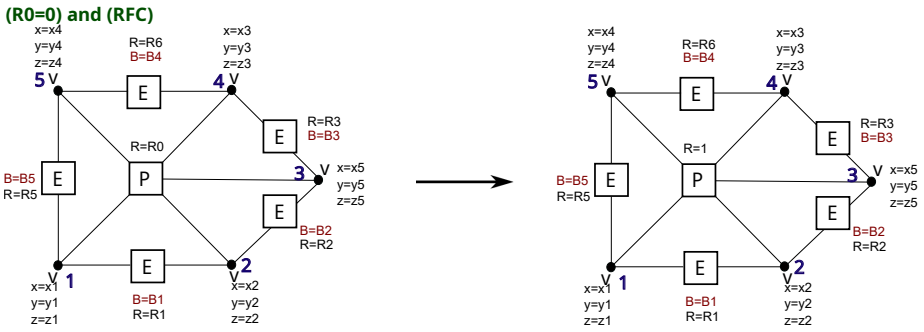

Fig. 13. Production **P6** - marking pentagon for breaking.

Production **P7**, shown in fig. 14, marks edges of **pentagonal element**, marked for refinement, for breaking - it sets value of attribute **R** of each hyperedge with label **E** to **1**

Production **P8** breaks the pentagonal element marked for refinement if all its edges are broken. Production removes the hyperedge **P** representing the "big" element and creates a new node in the center of this element. Production creates five new hyperedges with the label Q, representing the interiors of the newly created elements, and sets the values of the attribute **R** of the newly created

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

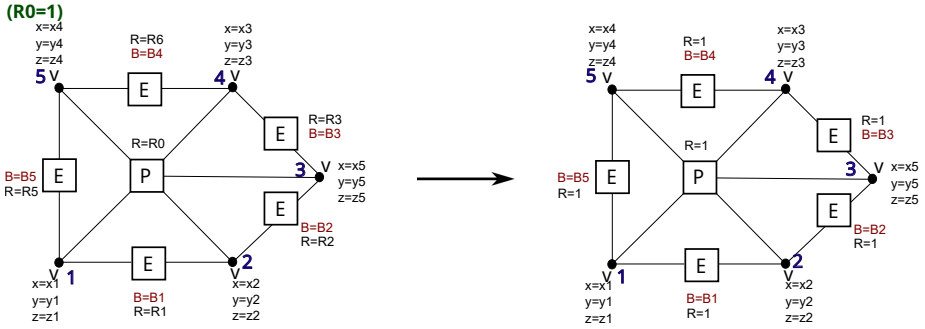

Fig. 14. Production **P7** - marking edges of pentagon for breaking.

hyperedges with the label **Q** to **0**. It also creates five new hyperedges with label **E** with $\mathbf{R} = \mathbf{0}$ and $\mathbf{B} = \mathbf{0}$. Production $P8$ is presented in figure 15.

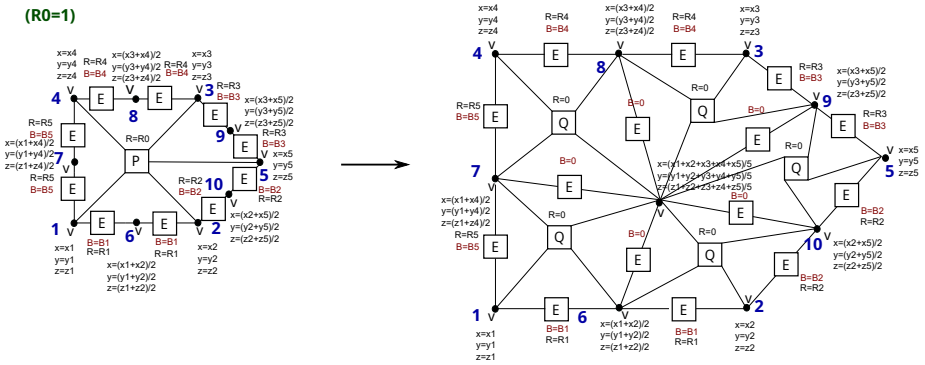

Fig. 15. Production **P8** breaking the pentagonal element marked for refinement.

Production **P9** marks an **hexagonal element** for refinement if it meets the refinement criterion - it sets the value of the attribute **R** of the hyperedge with the label **S** to **1**. Production **P9** is presented in fig. 16

Production **P10**, shown in fig. 17, marks edges of **hexagonal element**, marked for refinement, for breaking - it sets value of attribute **R** of each hyperedge with label **E** to **1**

Production **P11** breaks the hexagonal element marked for refinement if all its edges are broken. Production removes the hyperedge **S** representing the "big" element and creates a new node in the center of this element. Production creates six new hyperedges with the label Q, representing the interiors of the newly created elements, and sets the values of the attribute **R** of the newly created hyperedges with the label **Q** to **0**. It also creates six new hyperedges with label **E** with $\mathbf{R} = \mathbf{0}$ and $\mathbf{B} = \mathbf{0}$. Production $P11$ is presented in figure 18.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

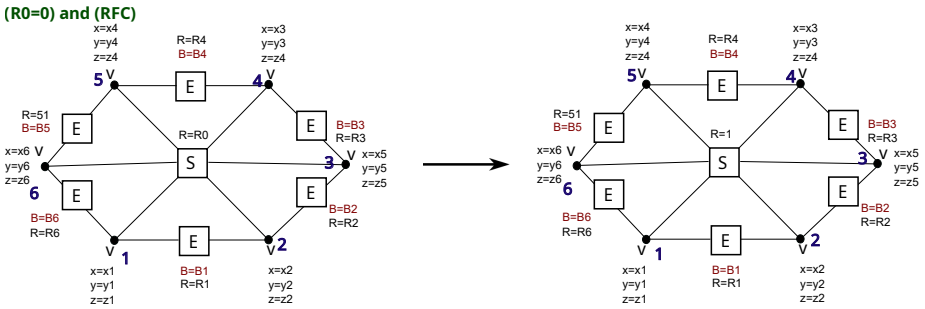

Fig. 16. Production **P9** - marking hexagon for breaking.

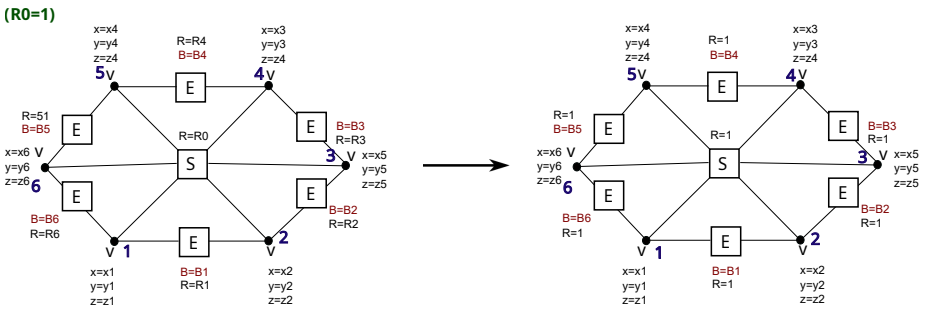

Fig. 17. Production **P10** - marking edges of hexagon for breaking.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

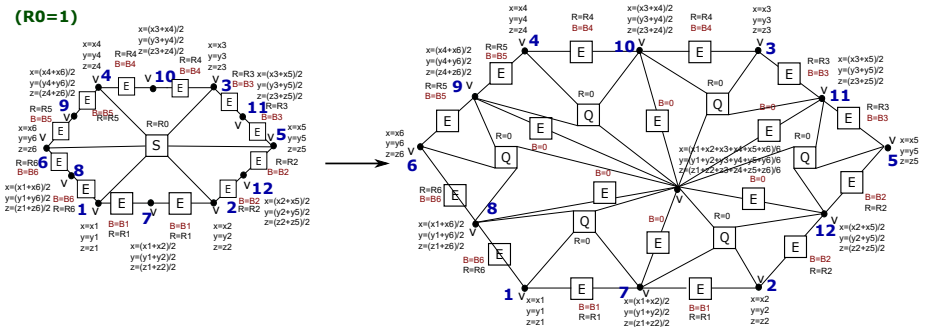

Fig. 18. Production **P11** - breaking the hexagonal element marked for refinement.

4 Exemplary derivation

The presented graph grammar was implemented in Python using NetworkX. The implementation utilizes a bipartite graph model within the NetworkX library, where nodes represent either vertices or hyperedges. The type of each graph node is defined by its label: Q, P, or S for nodes representing mesh elements (quadrilaterals, pentagons, and hexagons, respectively), and E for nodes representing edges of elements. The adaptation process is driven by the values of node attributes R (refinement flag) and B (boundary flag). Production Left-Hand Sides are identified through localized structural traversal, where subgraphs are matched by verifying node adjacency and the values of attributes. The matching strategy is based on iterating only over hyperedges with the appropriate label (pre-filtering) and local verification of the structure around each matching hyperedge. There is no need to analyze the entire graph — only elements directly connected to a given hyperedge.

Visualization of exemplary graph grammar derivation, following the idea of mesh refinements from figure 19, step by step, is presented in figures 20, 21. Visualization encodes the semantic state of the graph using color, line width, and symbolic labels. Hyperedges representing mesh elements are drawn as colored circular nodes and labeled with a single letter indicating their type: Q denotes a quadrilateral, P a pentagon, S a hexagon, and E an edge. For simplicity, the hyperedges with labels Q, P, and S, which conceptually connect all vertices belonging to the interior of the corresponding polygonal element, are not drawn with all nodes belonging to them. The color of an element reflects its refinement state: red colors indicate elements that are marked for breaking (attribute $R = 1$). Quadrilateral, pentagonal, and hexagonal hyperedges are shown in shades of blue, whereas edges are shown in shades of green. Edges are drawn as gray lines whose width encodes boundary information: thin lines represent internal edges (attribute $B = 0$), and thicker lines represent boundary edges (attribute $B = 1$).

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

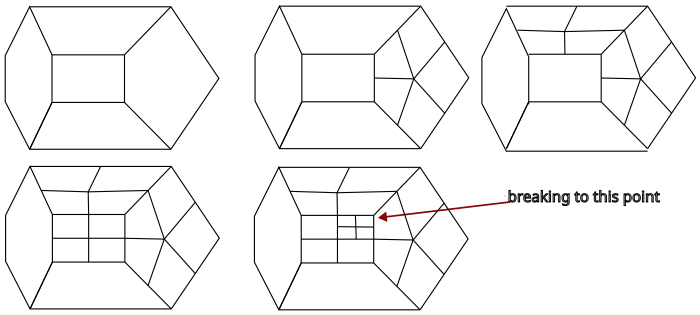

Fig. 19. The idea of mesh refinement according to the polygonal adaptation strategy of the mesh, step by step.

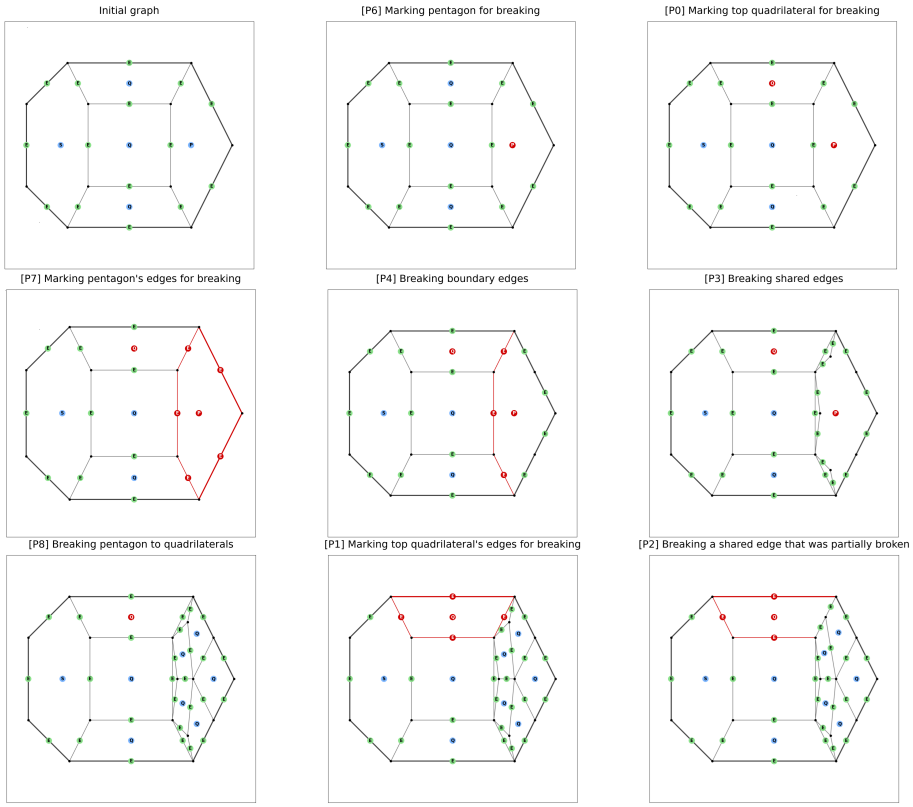

Fig. 20. Part 1 of hypergraph grammar derivation for the mesh from the figure 19 of the mesh, step by step.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

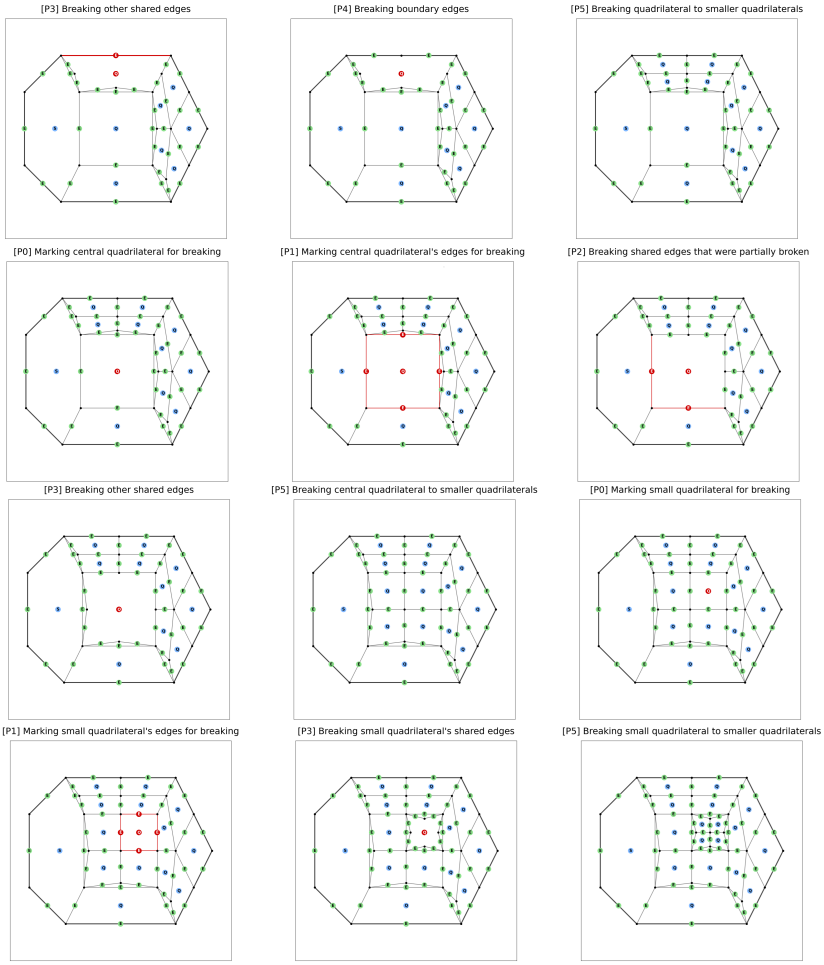

Fig. 21. Part 2 of hypergraph grammar derivation

5 Conclusion and future work

In the paper, the hypergraph grammar model for polygonal adaptive strategy, used in high-order polygonal discontinuous Petrov-Galerkin method, was introduced. The approach can be easily extended to other polygons, by adding productions analogous to productions $P0, P1$ and $P5$ for the corresponding n-gon. The hypergraph grammar presented was implemented in Python. The paper includes exemplary derivation performed on simple mesh using implemented code. Future work will involve parallelization of the introduced graph grammar model and using this mesh generator for performing numerical simulations.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11

6 Acknowledgements

This study was carried out as part of the fundamental research financed by the Ministry of Science and Higher Education, grant no. 16.16.110.663. The work of Albert Oliver-Serra and the visit of Anna Paszyńska has been funded by the grant contract "PRECOMP02 SD-24/03" (awarded by the Ministry of Universities, Science, Innovation, and Culture of the Government of the Canary Islands to the University of Las Palmas de Gran Canaria).

References

1. Hughes, TJR.: Linear static and dynamic finite element analysis. Dover Publications, (2000)
2. Demkowicz, L.: Computing with Hp-Adaptive Finite Elements, Vol. 1: One and Two Dimensional Elliptic and Maxwell Problems. Chapman & Hall / CRC Press (2006)
3. Astaneh, A., Fuentes, F., Mora, J., Demkowicz, L.: High-order polygonal discontinuous Petrov–Galerkin (PolyDPG) methods using ultraweak formulations. Computer Methods in Applied Mechanics and Engineering, **32**, 686-711 2018
4. Flasinski, M., Schaefer, R.: Quasi context sensitive graph grammars as a formal model of FE mesh generation. Computer-Assisted Mechanics and Engineering Science **3**, 191–203 (1996)
5. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part One: Realization of CP-Graphs, Machine Graphics and Vision **2(1)**, 3–38 (1993)
6. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part Two: CP-Graph. Grammars and Languages, Machine Graphics and Vision **2(2)**, 149–178 (1993)
7. Habel, A, Kreowski, HJ.: May We Introduce to You: Hyperedge Replacement. Lecture Notes in Computer Science **291** 5–26 (1987)
8. Habel, A., Kreowski, HJ.: Some Structural Aspects of Hypergraph Languages Generated by Hyperedge Replacement. Lecture Notes in Computer Science **247** 207-–219 (1987)
9. Fahmy, H, D. Blostein, B.:A survey of graph grammars: theory and applications. Proceedings., 11th IAPR International Conference on Pattern Recognition **2** 294–298 (1992)
10. Paszynska, A., Paszynski, M.: Graph transformations for modeling parallel hp-adaptive finite element method. Lecture Notes in Computer Science **4967** Berlin/Heidelberg 1313—1322 (2007)
11. Paszynska, A., Paszynski, M., Grabska, E.: Graph transformations for modeling hp-adaptive finite element method with mixed triangular and rectangular elements. Lecture Notes in Computer Science **5545** Berlin/Heidelberg 875—884 (2009)
12. Slusarczyk, G., Paszynska, A.: Hypergraph grammars in hp-adaptive finite element method. Procedia Computer Science **18** 1545—1554 (2013)
13. Podsiadło, K., Oliver Serra, A, Paszynska, A., Montenegro, R., Henriksen, I., Paszyński, M., Pingali, K.: Parallel graph-grammar-based algorithm for the longest-edge refinement of triangular meshes and the pollution simulations in Lesser Poland area. Engineering with Computers 1–24 (2021)
14. Paszyńska, A., Maczuga, P., Dobija, M., Oliver-Serra, A., Paruch, E., Radek, J.: Graph Grammar Model for H-Adaptation for Meshes with Quadrilateral, Pentagon, and Hexagon Elements. Lecture Notes in Computer Science Springer **1597** (2025)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_11