Procedia Computer Science

Volume 29, 2014, Pages 947–959

ICCS 2014. 14th International Conference on Computational Science

Dynamic Programming Algorithm for Generation of Optimal Elimination Trees for Multi-Frontal Direct Solver over h-Refined Grids

Hassan AbouEisha[1], Mikhail Moshkov[1], Victor Calo[1], Maciej Paszynski[2], Damian Goik[2], and Konrad Jopek[2]

[1] King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
hassan.aboueisha@kaust.edu.sa, mikhail.moshkov@kaust.edu.sa, victor.calo@kaust.edu.sa
[2] AGH University of Science and Technology, Krakow, Poland
paszynsk@agh.edu.pl, goik@student.agh.edu.pl, kjopek@gmail.com

Abstract

In this paper we present a dynamic programming algorithm for finding optimal elimination trees for computational grids refined towards point or edge singularities. The elimination tree is utilized to guide the multi-frontal direct solver algorithm. Thus, the criterion for the optimization of the elimination tree is the computational cost associated with the multi-frontal solver algorithm executed over such tree. We illustrate the paper with several examples of optimal trees found for grids with point, isotropic edge and anisotropic edge mixed with point singularity. We show the comparison of the execution time of the multi-frontal solver algorithm with results of MUMPS solver with METIS library, implementing the nested dissection algorithm.

Keywords: mesh adaptation, multi-frontal direct solver, elimination trees, dynamic programming

1 Introduction

In this paper we present a dynamic programming algorithm for finding optimal elimination trees for computational grids obtained from h adaptive finite element method [2]. The elimination tree [7] is a core part of the multi-frontal direct solver algorithm [3, 4, 5, 8], defining the order of elimination of nodes as well as the pattern for construction and merging of the frontal matrices, if the inputs for the solver algorithm are partially assembled element frontal matrices, not the fully assembled global problem. In other words the inputs for the multi-frontal solver algorithm are an elimination tree and element frontal matrices. Having the elimination tree we can estimate exactly the computational cost of the multi-frontal solver algorithm. Based on this principle we present a dynamic programming algorithm that constructs a class of elimination trees for a given mesh, and we select the optimal tree, namely the one that has minimum

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.085

computational cost estimate. This optimization algorithm can be utilized as a learning tool for the construction of heuristic algorithms for a class of refined grids.

The dynamic programming algorithm has been tested on several two dimensional grids, h-refined toward point or edge singularities. In particular we tested isotropic edge, isotropic point and anisotropic edge mixed with point singularities. The execution time of the multi-frontal solver algorithm using our optimal trees has been compared with MUMPS [1] solver with METIS [6] library.

We also presented how to generalize the results into three dimensional grids with analogous singularities.

2 Basic Notions

In this section, we describe the class of finite element meshes for which we construct optimal elimination trees and define the notion of an elimination tree. The class of meshes investigated is constructed as follows. We start with a regular grid that consists of a rectangle divided by horizontal and/or vertical straight lines into equal rectangular cells (see Figure 1). Those cells may be refined by two straight lines into four equal cells provided all of its sides are divisible. We define a side to be divisible iff it is a boundary side or its two endpoints are traversed by lines perpendicular to this side. A boundary side is a side that belongs to a border side of the initial rectangle. We can refine the new cells as long as all of their sides are divisible, etc.

The following example illustrates the previous process of construction and the discussed notions for the mesh presented in Figure 2. We begin with a regular grid $ABDE$ consisting of the two cells $ABCF$ and $CDEF$. $ABCF$ is refined next as all of its sides are boundary sides except CF which is divisible as lines AE and BD traverse its endpoints. Cell $HKJF$ cannot be refined next as the side JF is not divisible.

We define now the notion of dividing lines that are used to partition a given mesh. We denote the set of vertical lines that extend between the border sides of a mesh M by $S_V(M)$, horizontal lines that extend through the border sides of M by $S_H(M)$ and the union of both sets by $S(M)$. We do not consider vertical border sides of mesh M among $S_V(M)$ and similarly for its horizontal border sides. Mesh M is a unitary mesh iff it does not have any dividing lines, i.e., $S(M) = \phi$.

The mesh M can be partitioned using the dividing line l that belongs to the set $S(M)$. This partitioning step results in two submeshes: $M(l, 0)$ that represents the submesh which lies below (left of) horizontal (vertical) line l and $M(l, 1)$ denotes the submesh which is above (right of) the horizontal (vertical) line l. For each submesh M' we can define the notion of dividing lines in a similar way as for M, so we can use notation $S_V(M')$, $S_H(M')$ and $S(M')$.

In Figure 2, $S_V(ABDE) = \{GH, CF\}$, $S_H(ABDE) = \phi$ and $S(ABDE) = \{GH, CF\}$. $ABDE$ can be partitioned using the dividing line GH resulting in the two submeshes: $ABGH$ ($ABDE(GH, 0)$) and $GDEH$ ($ABDE(GH, 1)$) and it can also be partitioned using the dividing line CF.

We describe an arbitrary submesh of M by a sequence of partitioning steps. Formally, a submesh of a mesh M is an expression of the kind

$$M(l_1, \delta_1)(l_2, \delta_2)...(l_i, \delta_i)...(l_n, \delta_n)$$

where $\delta_1, \ldots \delta_n \in \{0, 1\}$, $l_1 \in S(M)$ and $l_i \in S(M(l_1, \delta_1)(l_2, \delta_2)...(l_{i-1}, \delta_{i-1}))$, for i = 2, ..., n.

The resulting submesh is described as follows. First, a dividing line l_1 is used to partition the mesh M then the submesh $M(l_1, \delta_1)$ is obtained. The line l_2 is a dividing line of this submesh

($l_2 \in S(M(l_1, \delta_1))$) which is used to partition $M(l_1, \delta_1)$ again until the desired submesh is obtained.

Let M' be a submesh of M and $l \in S(M')$. The line l represents a common border side between two submeshes: $M'(l, 0)$ and $M'(l, 1)$. We define $B(M')$ to be the number of edges on the border sides of M'. These edges result from lines that cut the border sides of M' or touching them if they are boundary sides. We consider $BE(M')$ as the number of edges that lie in the boundary sides. We denote by $E(l)$ the number of common edges of these two submeshes ($M'(l, 0)$ and $M'(l, 1)$) on l. For example, $B(ABDE) = 9$, $BE(BGKI) = 2$ and $E(GH) = 2$.

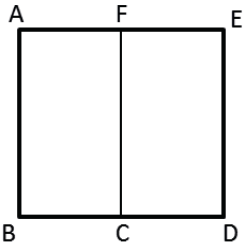

Figure 1: Regular mesh ABDE

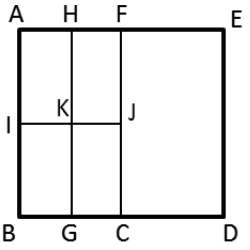

Figure 2: ABDE after refinement

An elimination tree describes a strategy for partitioning a finite element mesh. We define the notion of elimination tree for a mesh M by induction. Let M be a unitary mesh labeled with the identifier φ then there exists only one elimination tree for M presented in Figure 3. Let M be a nonunitary mesh (with dividing lines that can subdivide it). Then any elimination tree for M can be represented in the form given by Figure 4 where $l \in S(M)$ and τ_δ is an elimination tree for the submesh $M(l, \delta)$, $\delta = 0, 1$. We denote the set of all elimination trees for the mesh M by $P(M)$.

Let τ be an elimination tree for mesh M. This tree is a rooted binary tree. Any terminal node of such tree is labeled with an identifier of a unitary submesh of M. Any nonterminal node is labeled with a line. Each nonterminal node has exactly two edges that start from it and are labeled with 0 and 1, respectively. Figure 5 shows an example of an elimination tree for the mesh example presented in Figure 2.

We now associate to each node v of the elimination tree τ a submesh $M(v)$ of the mesh M. If v is the root of τ then $M(v) = M$. If v is not the root and in the path form the root to v nodes are labeled with lines $l_1, \ldots, l_m$ and edges are labeled with the numbers $\delta_1, \ldots, \delta_m$ then $M(v) = M(l_1, \delta_1) \ldots (l_m, \delta_m)$.

For each nonterminal node v of τ, this node is labeled with a dividing line from $S(M(v))$ which divides the submesh $M(v)$ into two submeshes. For each terminal node v, the submesh $M(v)$ is a unitary mesh and v is labeled with the identifier of $M(v)$.

3 Cost Functions

We can work with different cost functions for elimination trees which characterize time or space complexity of computations on different hardware platforms. Each cost function ψ is defined by induction on the pair (M, τ) where τ is an elimination tree for the mesh M. Let τ be a trivial elimination tree for a unitary mesh M. Then $\psi(M, \tau) = \psi^0(p, BE(M))$ where ψ^0 is an operator that maps a pair of nonnegative numbers to a nonnegative number. Let τ be an

Figure 3: Elimination tree for a unitary mesh

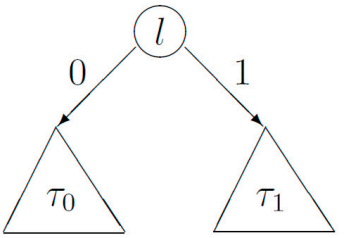

Figure 4: Elimination tree for a nonunitary mesh

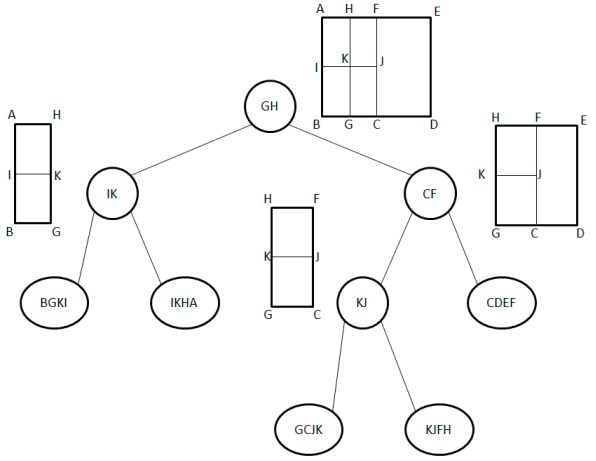

Figure 5: Elimination tree for the mesh example

elimination tree (presented in Figure 4) for a nonunitary mesh M that uses dividing line l as a first step in partitioning the mesh M. Then

$$\psi(M,\tau) = F(p, E(l), B(M), \psi(M(l,0),\tau_0), \psi(M(l,1),\tau_1))$$

where F is an operator that maps a 5-tuple of nonnegative numbers into a nonnegative number, and τ_0 and τ_1 are subtrees of τ (see Figure 4). The cost of partitioning the mesh M using its dividing line l depends on the global degree of approximation p, the parameters $E(l)$ and $B(M)$ in addition to the cost of the elimination trees of the resulting submeshes.[

A cost function is monotone if for any nonnegative numbers $a, b, c, d_1, d_2,\ e_1, e_2$ such that $d_i \leq e_i, i = 1, 2$, the inequality

$$F(a, b, c, d_1, d_2) \leq F(a, b, c, e_1, e_2)$$

holds, while, a strongly monotone cost function is a monotone cost function that for any nonnegative numbers $a, b, c, d_1, d_2,\ e_1, e_2,\ d_1 \leq e_1$ and $d_2 \leq e_2$, the inequality

$$F(a, b, c, d_1, d_2) < F(a, b, c, e_1, e_2)$$

holds if $d_i < e_i$ for some $i \in \{1, 2\}$.

In this paper, we consider the computational cost related to the sequential execution of the multi-frontal solver algorithm. The cost of processing a trivial elimination tree for a unitary mesh depends on the polynomial order of approximation p and is defined as follows:

$$\psi^0(p, BE(M)) = \sum_{i=1}^{p^2+BE(M)\times p} 3 \times (4p + 4 + i) \times (4p + 3 + i)$$

For a non-trivial elimination tree τ (presented in Figure 4) representing a nonunitary mesh M, the cost function depends on its children $c_0 = M(l, 0)$ and $c_1 = M(l, 1)$ and the line l used for partitioning as follows:

$$F(p, E(l), B(M), \psi(c_0, \tau_0), \psi(c_1, \tau_1)) = \psi(c_0, \tau_0) + \psi(c_1, \tau_1)$$
$$+ \sum_{i=1}^{E(l)\times(p+1)-1} 3 \times (B(M) \times (p + 1) + i) \times (B(M) \times (p + 1) + i - 1).$$

Thus, it can be shown that the considered cost function is strongly monotone.

4 Representation of the Set of Elimination Trees

We present an algorithm to construct the graph $G(M)$ for a mesh M. Nodes of this graph are submeshes of M.

We start with one node that represents the mesh M. Each node is then processed and marked. The algorithm terminates when all nodes are marked as processed. The work at each node η is done by considering each line l that belongs to the set $S(\eta)$. If $S(\eta) = \phi$, i.e., η is a unitary mesh, we mark it as processed and label it with its identifier. Otherwise, each line l produces a pair of edges, labeled with $(l, 0)$ and $(l, 1)$, connecting node η to nodes c_0 and c_1 respectively such that $c_0 = \eta(l, 0)$ and $c_1 = \eta(l, 1)$. If c_0 or c_1 are not present in the graph, nodes corresponding to them are created and connected to η. Any other node that is not marked is then chosen and the algorithm performs the steps described above. Nodes in this graph that lack any outgoing edges are called terminal nodes. These nodes correspond to unitary meshes.

It is clear that the previous graph is a directed acyclic graph (DAG). Now for each node η, we describe the set of elimination trees $P(\eta)$ corresponding to this node. Let η be a terminal node then η is a unitary mesh. The set of elimination trees corresponding to η ($P(\eta)$) contains only the trivial elimination tree consisting of one node labeled with φ denoting the identifier of this unitary mesh. Let η be a nonterminal node, we describe now an arbitrary elimination tree from $P(\eta)$. Let $l \in S(\eta)$, $\tau_0 \in P(\eta(l, 0))$ and $\tau_1 \in P(\eta(l, 1))$. We describe an elimination tree τ of the set $P(\eta)$ as follows: τ has a root labeled with the dividing line l and a pair of edges are directed from this root towards the elimination trees τ_0 and τ_1. Those edges are labeled with 0 and 1, respectively.

Lemma 1 Let M be a mesh and η be a node in the graph $G(M)$, then the set $P(\eta)$ coincides with the set of all elimination trees for the mesh η.

Proof We prove this proposition by induction on the nodes of $G(M)$. Let η be a terminal node in this graph. In this case η is a unitary mesh labeled with identifier φ. Then, there exists only one elimination tree consisting of one node labeled with φ and this tree is included in $P(\eta)$. Therefore the statement of lemma holds for η.

We show that any elimination tree τ for η is included in the set $P(\eta)$. Consider a non-terminal node η and assume that the statement holds for all its descendants. Let τ be an arbitrary elimination tree corresponding to η. It is clear that τ consists of more than one node as η is not a unitary mesh. Consider line l to be the root of τ, τ_0 to be the subtree connected to the root with edge labeled with 0 and τ_1 to be the other subtree. It follows from the definition of the trees corresponding to a non-terminal node η that line l belongs to the set $S(\eta)$, τ_0 and τ_1 corresponds to nodes $\eta(l,0)$ and $\eta(l,1)$ respectively. According to the inductive hypothesis, the trees τ_0 and τ_1 are included in the sets $P(\eta(l,0))$ and $P(\eta(l,1))$, respectively. Therefore, the tree τ is an element of $P(\eta)$.

It can be shown that any elimination tree $\tau \in P(\eta)$ is an elimination tree for η. The root of τ is a line l that belongs to the set $S(\eta)$. Moreover, the root has two outgoing edges: one labeled with 0 entering to subtree τ_0 and other labeled with 1 entering to subtree τ_1. According to the method of construction of $P(\eta)$, $\tau_0 \in P(\eta(l,0))$ and $\tau_1 \in P(\eta(l,1))$. By the inductive hypothesis, τ_0 and τ_1 corresponds to $\eta(l,0)$ and $\eta(l,1)$. Consequently, the tree τ is an elimination tree for η.

5 Optimization Procedure

In this section, we describe the notion of a proper subgraph of $G(M)$. We show how this notion is used during the optimization procedure.

5.1 Proper Subgraph of $G(M)$

Consider the graph $G(M)$. For each non-terminal node of $G(M)$, we can remove any pair of edges going out of this node under the condition that at least one pair of edges leaves this node. Each subgraph of $G(M)$ that is constructed as described is considered a proper subgraph of $G(M)$. The set of elimination trees corresponding to a node η of a proper subgraph G' can be defined in a similar way as for $G(M)$. We denote by $P_{G'}(\eta)$ the set of elimination trees corresponding to a node η of the proper subgraph G'.

5.2 Optimization Process

Let $G = G(M)$, we present now the procedure of optimization of elimination trees relative to the cost function ψ defined by the operators ψ^0 and F. The optimization procedure is performed bottom up. To begin, all the terminal nodes v of G are assigned the cost $\psi^0(p, BE(v))$. During the procedure of optimization, we label each nonterminal node η with a number $C(\eta)$ which is equal to the minimum cost of the elimination tree for η relative to ψ and we remove some pair of edges issuing from η.

Let η be a non-terminal node. We consider an arbitrary pair of edges issuing from η and labeled with a pair $(l,0)$ and $(l,1)$ where $l \in S(\eta)$. Those edges enter nodes $c_0 = \eta(l,0)$ and $c_1 = \eta(l,1)$ labeled with numbers $C(c_0)$ and $C(c_1)$. We correspond to this pair the number $C(\eta,l) = F(p, E(l), B(\eta), C(c_0), C(c_1))$. We define $C(\eta) = \min\{C(\eta,l) : l \in S(\eta)\}$. After that we remove all pair of edges issuing from η and labeled with $(l,0)$ and $(l,1)$ for which $C(\eta,l) > C(\eta)$. In other words, we remove all pairs of edges that result in elimination trees with cost greater than the optimal elimination tree for η.

Once we perform the previous work on all the nodes, we obtain the proper subgraph G_ψ. We denote the set of elimination trees corresponding to a node η in G_ψ by $P_{G_\psi}(\eta)$.

Let M be a mesh and ψ be a cost function. Let $G = G(M)$ and η be an arbitrary node in G. We denote by $P^{opt}_{\psi,G}(\eta)$ the subset of $P_G(\eta)$ that contains all elimination trees having minimum value with respect to the cost function ψ, i.e., $P^{opt}_{\psi,G}(\eta) = \{\hat{\tau} \in P_G(\eta) : \psi(\eta, \hat{\tau}) = \min_{\hat{\tau} \in P_G(\eta)} \psi(\eta, \hat{\tau})\}$. This set of elimination trees corresponds to the actual set of optimal elimination trees for a given node η.

Theorem 1 and Theorem 2 describe important properties of the set $P_{G_\psi}(\eta)$ for the cases of monotone and strongly monotone cost functions.

Lemma 2 Let M be a mesh and ψ be a monotone cost function defined by the pair of operators (ψ_0, F). Consider G equal to $G(M)$, η to be an arbitrary node in G and $C(\eta)$ be the value assigned to η during optimization procedure. Then for any elimination tree τ from the set $P_{G_\psi}(\eta)$, $\psi(\eta, \tau) = C(\eta)$.

Proof

We prove this lemma by induction. Let η be a terminal node in G, then there exists only one elimination tree (depicted in Figure 3) and the statement holds.

Let η be a non-terminal node in G and the statement holds for all the descendants of η. Consider τ to be an arbitrary elimination tree (presented in Figure 4) that belongs to the set $P_{G_\psi}(\eta)$. Let $c_0 = C(\eta(l, 0))$ and $c_1 = C(\eta(l, 1))$ be the values assigned to nodes $\eta(l, 0)$ and $\eta(l, 1)$. According to the procedure of optimization, $\psi(\eta, \tau) = F(p, E(l), B(\eta), c_0, c_1)$. By the induction hypothesis, $c_0 = \psi(\eta(l, 0), \tau_0)$ and $c_1 = \psi(\eta(l, 1), \tau_1)$.

Combining these equations, we have the following:

$$\begin{aligned} \psi(\eta, \tau) &= F(p, E(l), B(\eta), \psi(\eta(l, 0), \tau_0), \psi(\eta(l, 1), \tau_1)) \\ &= F(p, E(l), B(\eta), c_0, c_1) \\ &= C(\eta). \end{aligned}$$

Theorem 1 Let M be a mesh and ψ be a monotone cost function. Consider G equal to $G(M)$ and η to be an arbitrary node in G. Then we have $P_{G_\psi}(\eta) \subseteq P^{opt}_{\psi,G}(\eta)$.

Proof We prove this theorem by induction. Consider η to be a terminal node of G, then it is clear that there is only one elimination tree for η (depicted in Figure 3). This tree is the only member of $P_{G_\psi}(\eta)$ as well as $P^{opt}_{\psi,G}$ so the statement holds.

Let η be a non-terminal node of G. Assume that the statement is true for all descendants of η. Let $c = C(\eta)$ be the cost associated with node η during procedure of optimization. All elimination trees belonging to $P_{G_{\psi(\eta)}}$ have cost c as indicated in Lemma 1.

Consider an arbitrary elimination tree τ that belongs to the set $P^{opt}_{\psi,G}$. It is presented in Figure 4 as it is associated with non-terminal node η. We prove our statement by proving that $\psi(\eta, \tau) = c$. As $\tau \in P^{opt}_{\psi,G}$, $\psi(\eta, \tau) \leq c$. It is clear that τ_j belongs to the set $P_G(\eta(l, j))$ where $j \in \{0, 1\}$. Let c_j be a number assigned to node $\eta(l, j)$ during the procedure of optimization, then $\psi(\eta(l, j), \tau_j) \geq c_j$. Since ψ is a monotone cost function, we have the following:

$$\begin{aligned} \psi(\eta, \tau) &= F(p, E(l), B(\eta), \psi(\eta(l, 0), \tau_0), \psi(\eta(l, 1), \tau_1)) \\ &\geq F(p, E(l), B(\eta), c_0, c_1) \\ &\geq c. \end{aligned}$$

From the last two inequalities and $\psi(\eta, \tau) \leq c$, we have $\psi(\eta, \tau) = c$.

Theorem 2 Let M be a mesh and ψ be a strongly monotone cost function. Consider $G = G(M)$ and η to be an arbitrary node in G. Then $P_{G_\psi}(\eta) = P^{opt}_{\psi,G}(\eta)$.

Proof Since ψ is a strongly monotone cost function, it is also a monotone cost function and $P_{G_\psi}(\eta) \subseteq P^{opt}_{G,\psi}(\eta)$. So we need to prove that $P^{opt}_{\psi,G}(\eta) \subseteq P_{G_\psi}(\eta)$. We prove this by induction. Let η be a terminal node in G, then there is only one elimination tree (depicted in Figure 3) for η that is a member of both sets. Therefore $P^{opt}_{\psi,G}(\eta) \subseteq P_{G_\psi}(\eta)$.

Let η be a non-terminal node in G and the statement holds for all descendants of η. Consider an arbitrary elimination tree τ from $P^{opt}_{\psi,G}(\eta)$ presented in Figure 4. The trees τ_0 and τ_1 belong

to the sets $P^{opt}_{G,\psi}(\eta(l,0))$ and $P^{opt}_{G,\psi}(\eta(l,1))$ respectively as ψ is a strongly monotone cost function. By the inductive hypothesis, τ_0 and τ_1 belongs to the sets $P_{G_\psi}(\eta(l,0))$ and $P_{G_\psi}(\eta(l,1))$ respectively. Consider the pair of edges leaving node η in G and labeled with $(l,0)$ and $(l,1)$. This pair exists also after optimization procedure as $\tau \in P^{opt}_{G,\psi}(\eta)$. Then the elimination tree τ also belongs to $P_{G_\psi}(\eta)$.

6 Numerical results

This section contains numerical results concerning generation of optimal elimination trees for point, edge and point plus edge singularities together with comparison to nested dissection heuristic algorithm implemented by METIS library.

6.1 Point Singular Mesh

We denote by point singular meshes those meshes having one of their vertices as the point of singularity. For each refinement step, all base meshes that are closest to this vertex will be refined. Without loss of generality, we consider this vertex to be the bottom left vertex of the mesh in our experiments. An example of such meshes is presented in Figure 6. This mesh has vertex E as the point of singularity and it has been refined three times.

The two dimensional elimination tree results in the multi-frontal solver algorithm that eliminates elements surrounding the point singularity level by level. This algorithm can be generalized into three dimensional grids with point singularities. The resulting scalability of the solver algorithm is linear, compare Figure 10.

6.2 Point and Edge Singularity

The position of singularity in this type of mesh is one of its vertices. In each refinement step, all unitary meshes closest to this vertex are refined. However, the vertical line used in refinement extends through the whole initial mesh instead of the refined unitary mesh only. We consider in these experiments the vertex of singularity to be the bototm right vertex. An example of this mesh is presented in Figure 7 and it has been refined three times.

6.3 Edge Singular Mesh

Meshes with edge singularity have an edge as an object of singularity. For each refinement step, all base meshes that are closest to this edge will be refined. Without loss of generality, we consider this edge to be the bottom edge of the mesh. An example of such meshes is presented in Figure 8. This mesh has edge AE as the singular edge and there has been two refinement steps.

Moreover, for the edge singularity case we have compared the execution time of the multi-frontal solver using our optimal elimination tree with the MUMPS solver [1] interfaced with METIS [6] library, implementing the nested dissection algorithm. The comparison is presented in Figure 9. From the comparison it implies that our elimination tree reduces the execution time by order of magnitude. For the other singularities (point and point plus edge) the execution time of the solver based on our optimal trees is equal to the execution time of MUMPS solver with METIS library.

The two dimensional elimination tree results in the eliminates of elements, followed by elimination of patches of elements with increasing size. The algorithm can be also generalized

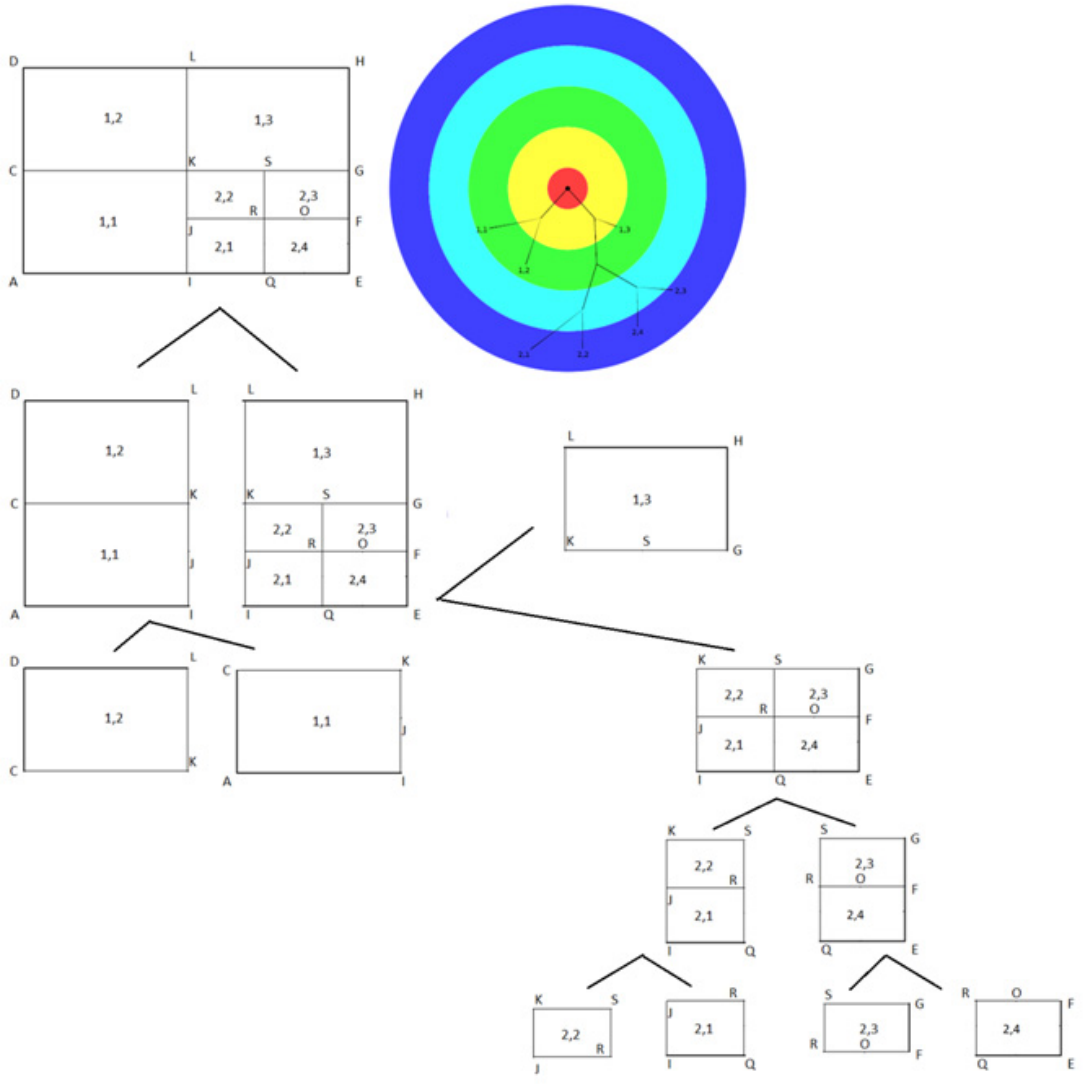

Figure 6: Optimal elimination tree for point singularity

into three dimensions. The resulting scalability of the solver algorithm is also linear, compare Figure 11.

7 Conclusions

In this paper we presented a dynamic programming algorithm allowing the construction of optimal elimination trees for grids refined towards point, edge or mixed point plus edge singularities. We showed that the execution time of the multi-frontal direct solver based on our optimal elimination tree for edge singularity is one order of magnitude faster than MUMPS

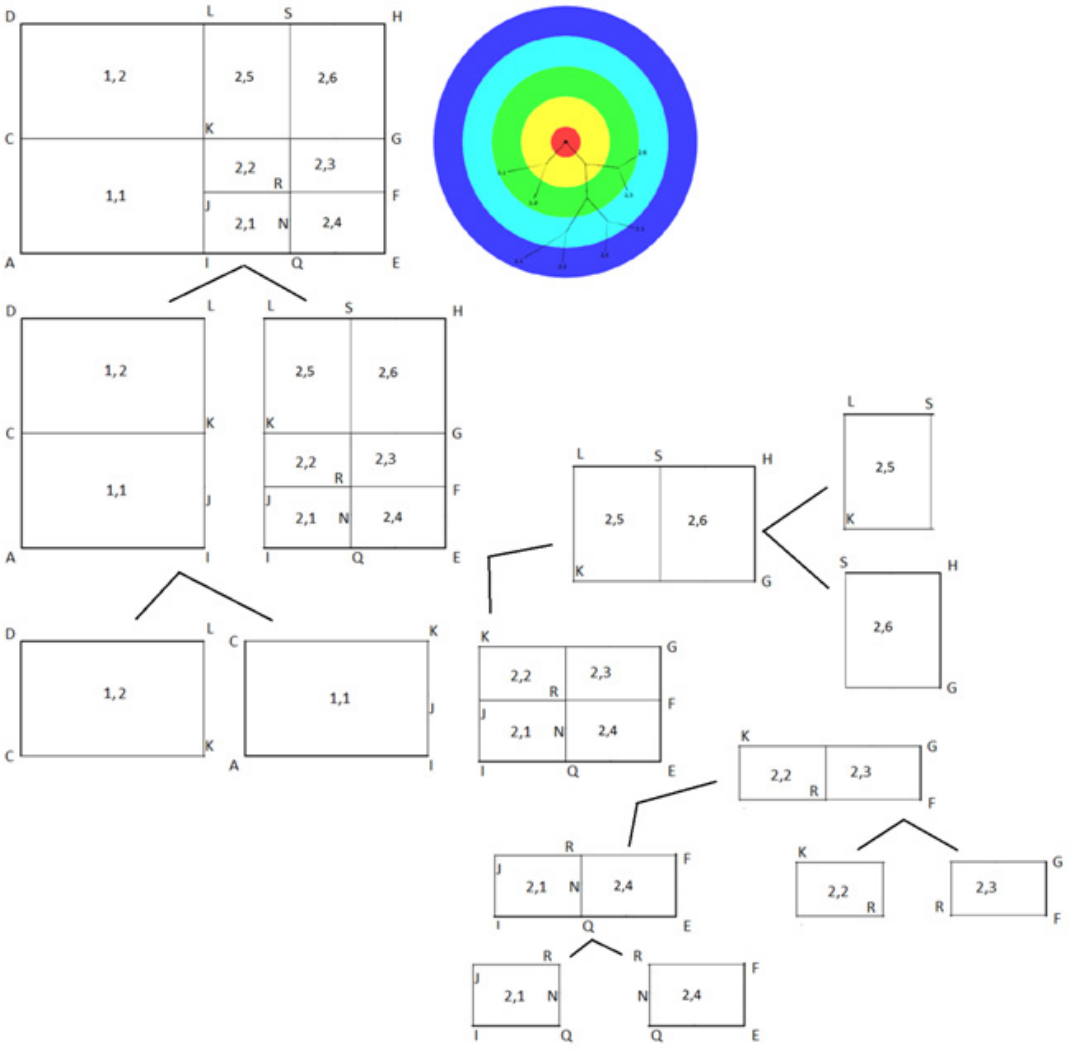

Figure 7: Optimal elimination tree for point plus anisotropic edge singularity

solver with METIS library. We also generalized the resulting elimination trees into three dimensional cases, and we showed the linear computational cost of the solver algorithm.

Acknowledgments The work presented in this paper has been supported by Polish MNiSW grant no. 2012/07/B/ST6/01229.

References

[1] Patrick R Amestoy, Ian S Duff, and J-Y L'Excellent. Mumps multifrontal massively parallel solver version 2.0. 1998.

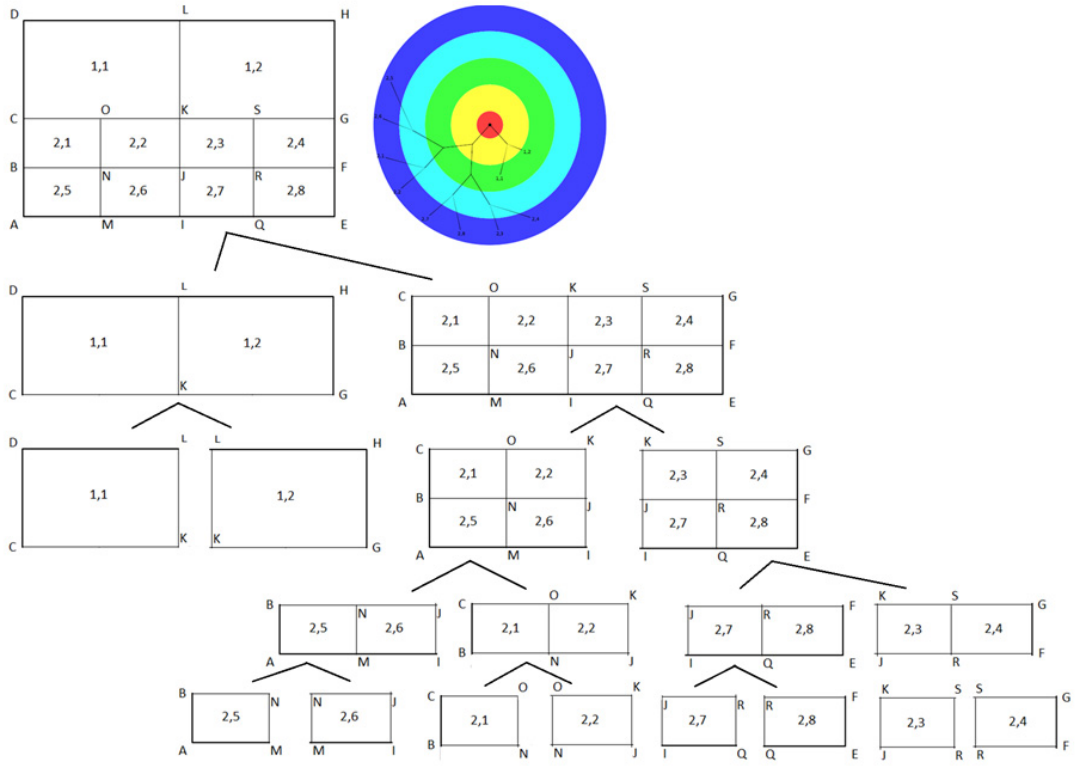

Figure 8: Optimal elimination tree for edge singularity

[2] Leszek Demkowicz. *Computing with hp Adaptive Finite Element Method: Volume 1 One and Two Dimensional Elliptic and Maxwell problems*. CRC Press, 2006.

[3] Iain S Duff and John K Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Transactions on Mathematical Software (TOMS)*, 9(3):302–325, 1983.

[4] Iain S Duff and John K Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal on Scientific and Statistical Computing*, 5(3):633–641, 1984.

[5] P Geng, JT Oden, and RA Van de Geijn. A parallel multifrontal algorithm and its implementation. *Computer Methods in Applied Mechanics and Engineering*, 149(1):289–301, 1997.

[6] George Karypis and V Kumar. Metis-serial graph partitioning and fill-reducing matrix ordering, 2012.

[7] Joseph WH Liu. The role of elimination trees in sparse factorization. *SIAM Journal on Matrix Analysis and Applications*, 11(1):134–172, 1990.

[8] Maciej Paszyński and Robert Schaefer. Graph grammar-driven parallel partial differential equation solver. *Concurrency and Computation: Practice and Experience*, 22(9):1063–1097, 2010.

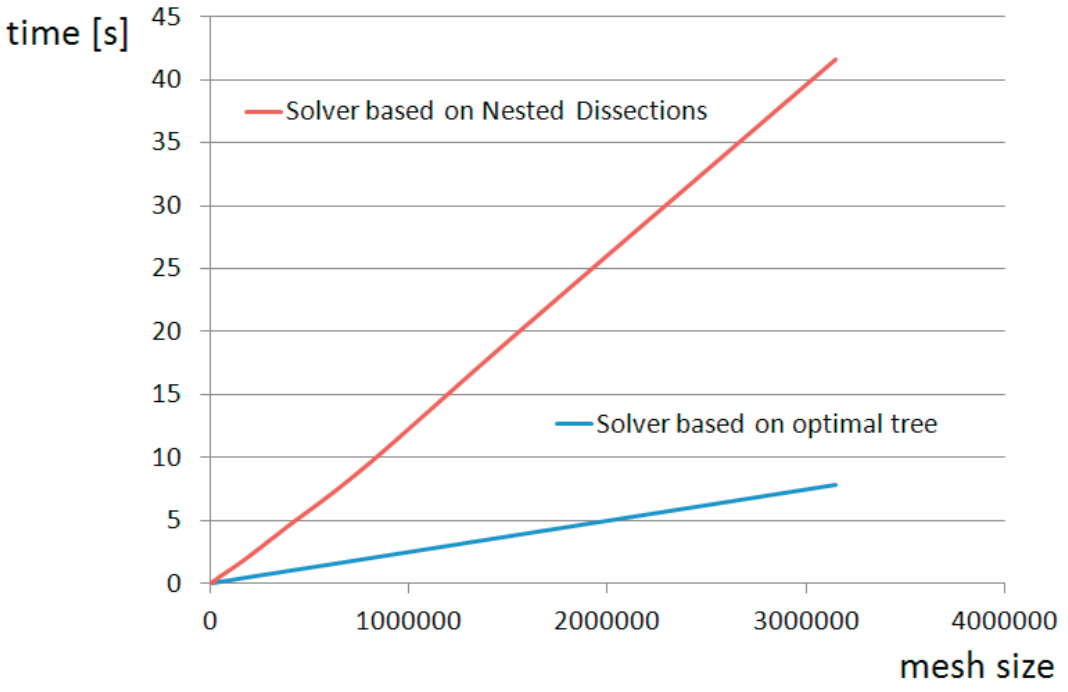

Figure 9: Execution time of the solver algorithm based on the optimal tree for two dimensional mesh in comparison with the MUMPS execution time

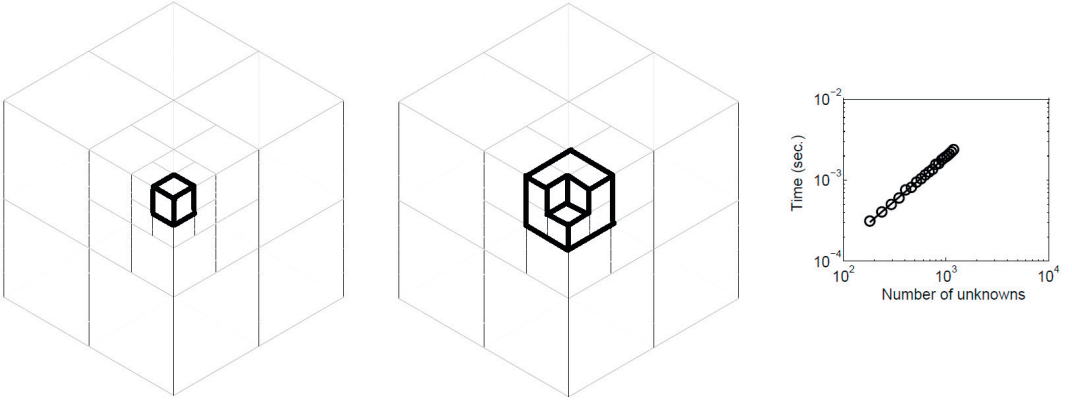

Figure 10: Elimination pattern for the three dimemesional point singularity and the resulting scalability of the solver algorithm

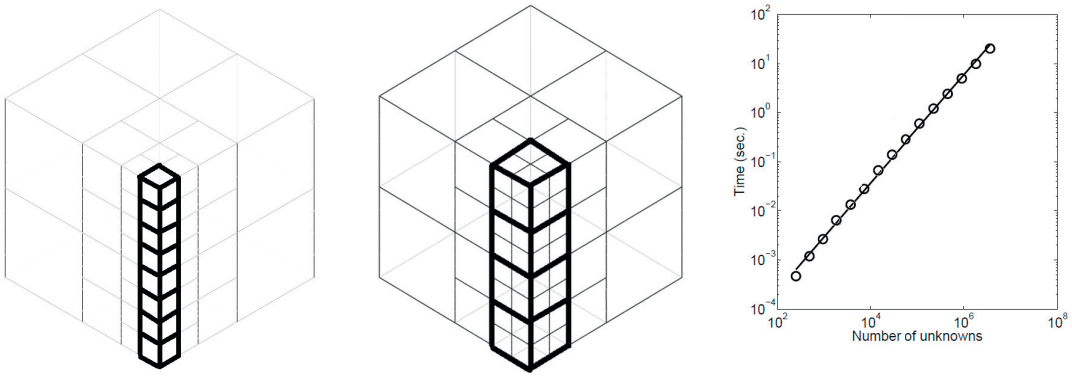

Figure 11: Elimination pattern for the three dimemesional edge singularity and the resulting scalability of the solver algorithm