Computers and Structures 265 (2022) 106770

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Efficient distributed approach for density-based topology optimization using coarsening and h-refinement

David Herrero-Pérez [a,b,*], Sebastián Ginés Picó-Vicente [a,b], Humberto Martínez-Barberá [a,c]

[a] Computational Mechanics and Scientific Computing Group, Technical University of Cartagena, 30202 Cartagena, Murcia, Spain
[b] Technical University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Murcia, Spain
[c] University of Murcia, Computer Science Faculty, 30100, Murcia, Spain

ARTICLE INFO

Article history:
Received 7 September 2021
Accepted 10 February 2022
Available online 3 March 2022

Keywords:
Topology optimization
Distributed computing
Multigrid methods
Adaptive mesh refinement

ABSTRACT

This work presents an efficient parallel implementation of density-based topology optimization using Adaptive Mesh Refinement (AMR) schemes to reduce the computational burden of the bottleneck of the process, the evaluation of the objective function using Finite Element Analysis (FEA). The objective is to obtain an equivalent design to the one generated on a uniformly fine mesh using distributed memory computing but at a much cheaper computational cost. We propose using a fine mesh for the optimization and a coarse mesh for the analysis using coarsening and refinement criteria based on the thresholding of design variables. We evaluate the functional on the coarse mesh using a distributed conjugate gradient solver preconditioned by an algebraic multigrid (AMG) method showing its computational advantages in some cases by comparing with geometric multigrid (GMG) and AMG methods in two- and three-dimensional problems. We use different computational resources with small regularization distances for such comparisons. We also evaluate the performance and scalability of the proposal using a different number of computing cores and distributed computing hosts. The numerical results show a significant increment of the computing performance for the overall computing time of the proposal combining dynamic coarsening, adaptive mesh refinement, and distributed memory computing architectures.

© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Topology optimization aims to find the optimal distribution of material within a design domain by minimizing a cost function subjected to a set of constraints. It is a powerful tool for engineers and scientists providing innovative and high-performance conceptual designs at the early stages of the design process without assuming any prior structural configuration. For this reason, it applies to a broad spectrum of applications [1,2]. There exist several topology optimization methods using different representations to describe the shapes they involve. Density-based topology optimization operates on a fixed mesh of finite elements penalizing the mechanical properties of elements. This penalization uses an interpolation function to find the optimal void/solid material distribution that minimizes an objective function. The homogenization method [3] and the Solid Isotropic Material Penalization (SIMP) method [4,5] are the most popular density-based topology optimization approaches, being the latter the most implemented method in commercial software probably due to its simplicity and feasibility. We can mention the implementation of the SIMP method in *OptiStruct* [6–8] by Altair Engineering, *Tosca Structure* [9,10] by Dassault Systems, *MSC Nastran* [11] by Hexagon, *Genesis* [12] by Vanderplaats Research and Development, and *Comsol Multiphysics* [13] commercial software, to name but a few.

Density-based topology optimization models the mechanical properties of finite elements using a power-law interpolation function between void and solid. The interpolation function relates the design variable and Young's modulus of finite elements. This strategy allows us to use gradient-based optimization methods to address the optimization problem. The number of optimization variables is of paramount importance in the optimization process. In the ideal case, the refinement of the mesh refinement increases the accuracy of the Finite Element Analysis (FEA) and improves the evaluation of the objective function. We usually capture more details in the optimized design by increasing the number of design variables obtaining an improved description of the boundaries between the empty and the solid finite elements. For these reasons,

* Corresponding author at: Structures and Construction Department, Technical University of Cartagena, Campus Muralla del Mar, 30202 Cartagena, Murcia, Spain.
E-mail addresses: david.herrero@upct.es (D. Herrero-Pérez), sebastian.pico@upct.es (S.G. Picó-Vicente), humberto@um.es (H. Martínez-Barberá).

https://doi.org/10.1016/j.compstruc.2022.106770
0045-7949/© 2022 The Author(s). Published by Elsevier Ltd.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

we usually obtain more optimized designs by increasing the number of design variables. However, the use of ersatz material approximation introducing a weak phase mimicking void material induces errors in the FEA [14] but avoids the singularity of the rigidity matrix. The tessellation cannot be too small since the problem becomes more and more badly conditioned as the discretization size tends to zero [15]. This lack of precision can be an obstacle to the optimization problem. Besides, the use of high-resolution finite element models gives rise to a high-size system of equations. Such systems should be solved efficiently to make the process feasible, both in computing time and computational resource requirements. The resolution of large systems of equations is a well-known computational challenge induced by the constant-increasing in the required fidelity and complexity of finite element models [16].

We can adopt different strategies to improve the computational efficiency of density-based topology optimization. We can mention the rescaling of large systems of equations to reduce the ill-conditioning [17], the use of low accurate approximations [18] of the analysis solution, and the use of efficient preconditioners [19]. The reanalysis strategies that avoid the analysis of the modified design in the optimization procedure are also rewarding, such as the approximate reanalysis [20] by only solving the system of equations at an interval of iterations and approximating the solution at other iterations of the nested topology optimization process. We also have to mention the multiresolution schemes, which decouple analysis and design discretizations. They improve the computational performance by using a coarse discretization for the analysis and a fine discretization for the design variables [21,22]. However, the iterative updates of the topology optimization design variables depend on the analysis results, and the order and type of finite elements limit the maximum resolution of the design variables [23]. Finally, we also can improve the computational performance using High-Performance Computing (HPC) to address the computationally intensive tasks of the topology optimization process using multi-core [24–28] and many-core [29–33] computing.

The Adaptive Mesh Refinement (AMR) technique [34] is a powerful tool to save computational cost and reduce the error in the estimated behavior by increasing and decreasing the number or order of the finite elements needed in the regions of interest. We determine such areas of interest using an error estimator, which indicates us either coarsening the unnecessary finite elements or refining them in the areas of interest. We can use different error estimators for the analysis and the design discretization [35] in topology optimization because the former aims to increase the accuracy of the magnitude to estimate, whereas the latter deals with the performance of the optimization. The error estimators for the analysis in topology optimization can be motivated to improve the estimation of some magnitude used in the problem formulation, such as the accurate calculation of stresses in stress-constrained topology optimization using AMR [36,37]. However, the improvement of the estimation of the displacement field used in minimum compliance and mechanism design problems do not modify the final design meaningfully. Indeed, the use of low accurate approximations [18] of the analysis solution is one technique often used to improve the computational efficiency of density-based topology optimization. We only require error estimators for the optimization to obtain an equivalent design to the one generated on a uniformly fine mesh but at a much cheaper computational cost [38]. We can achieve significant computational improvements when the topology optimization requires relatively small volume fractions since the design domain is highly void [39] after a few iterations of the optimization process. The computation of regions with weak material contributes significantly to the overall computational cost but little to the accuracy of the optimized design.

The early work of Kikuchi et al. [40] introduced adaptive grid methods combined with automatic re-meshing applied to shape optimization design problems. It aims to reduce the error associated with the finite element discretization avoiding distortion near the design boundaries. Costa and Alves [41] combine the topology optimization and the h-adaptive finite element methods to bound the error of analysis, improve the definition of the material boundary, and improve the performance by reducing the number of design variables. It adopts a recursive strategy assuming the converged solution on a coarse mesh to refine it and starting a new optimization on the fine mesh. They do not revisit coarse meshes after generating the new fine mesh. Stainko [42] only refines the material on the boundary indicated by the regularized intermediate density control method [43]. However, the optimization in the coarse mesh and then refining to optimize again in the fine mesh generates a smoother solution of the topology optimization in the coarse mesh instead of the counterpart solution in the fine mesh. In other words, the topology optimization in the fine mesh can be quite different from the solution refining the design obtained in the coarse meshes. Thus, these strategies may lead to suboptimal designs. For these reasons, De Sturler et al. [39] introduce the coarsening in the adaptive method to generate a similar result to the one generated on a uniformly fine mesh. We can optimize the performance of the dynamic refinement by only performing the coarsening inside the void region and at the last adaptive step [44].

Recent topology optimization works use dynamic AMR with parallel computing to address high-resolution minimum compliance problems. Liu et al. [45] propose a shared-memory adaptive topology optimization framework using the SIMP method restricting the simulation of elastic deformation to a narrow band surrounding the high-density region. They use a matrix-free implementation of the conjugate gradient solver preconditioned with a geometric multigrid (GMG) method for uniform grids, which permits them to obviate the computational effort on large void regions on the fly. The use of uniform grids is attractive in topology optimization applications due to its multiple advantages in computation [27,32]. These advantages include cache-coherent memory access, regular subdivisions for the parallelization of simple data layout, and the existence of efficient numerical PDE solvers using computational patterns, such as stencil-based computation. The use of uniform grids can require the accurate representation of the geometry from the CAD model, which can be estimated on the corresponding bounding box using AMR techniques [46,47]. However, a Cartesian grid enclosing the CAD model usually requires many more finite elements than tessellating the CAD model directly. Li et al. [48] propose a distributed-memory adaptive topology optimization framework using the level-set method for evolving a high-resolution unstructured mesh. It uses algebraic multigrid (AMG) methods for preconditioning the Krylov subspace iteration solver for evaluating the objective function and the reaction–diffusion equation used for the level-set update. The proposal achieves clear and high-resolution solid-void material boundary reducing the computational cost by coarsening the mesh distributed in the void domain.

This work aims to reduce the computational cost of density-based topology optimization by combining dynamic AMR and distributed computing. The objective is to obtain an equivalent design to the one generated on a uniformly fine mesh using distributed memory computing but at a much cheaper computational cost. We focus on reducing the computational cost of the bottleneck of the topology optimization process, the FEA for evaluating the objective function [16,33], providing an equivalent design to using the fine mesh. We use a fine mesh for the optimization and a

dynamic coarser mesh for the analysis using local coarsening and refinement criteria based on the thresholding of indicators based on design variables, such as the gradient of compliance or the own design variables. We perform the regularization, computation of sensitivities, and optimization using the fine mesh, whereas we use the dynamic coarse mesh for the analysis. The AMR strategy requires adaptive coarsening and refinement up to the fine mesh at all topology optimization iterations. This adaptive approach ensures obtaining a similar result than optimizing using the fine mesh at the cost of increasing the computational cost. Thus, we have to find a trade-off between the cost of solving and the computational requirements to reduce the system of equations to solve. We also avoid using computational patterns, which reduce the tessellation size to the geometry of the model to optimize.

There exists a consensus in the scientific community that Krylov subspace iteration solvers preconditioned with multigrid methods are the most efficient techniques to solve large systems of equations in topology optimization [26,45,48]. We adopt a distributed conjugate gradient solver preconditioned by an AMG for solving using the dynamic AMR technique proposed by Červený et al. [49]. We evaluate the performance of the distributed conjugate gradient solver preconditioned using different multigrid methods: AMG and GMG. We perform this comparison using diverse computational resources and regularization sizes. We also evaluate the performance and scalability of the proposal in two- and three-dimensional topology optimization problems using different computational resources. The numerical results show the benefits in performance using dynamic AMR for addressing big topology optimization problems using reduced computational resources.

We organize the remainder of the paper as follows. Section 2 reviews the basis and theoretical background of density-based topology optimization. We devote Section 3 to the introduction of adaptive mesh refinement and coarsening methods. Section 4 presents the distributed framework and the required communications system for calculating the computational resources together. Section 5 shows the numerical experiments evaluating the performance, feasibility, and scalability of the techniques adopted for taking advantage of distributed multi-core computing in density-based topology optimization. Finally, Section 6 presents the conclusion of the proposal.

2. Density-based topology optimization

Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain whose boundary is decomposed into three disjoint parts $\partial\Omega = \Gamma_u \cup \Gamma_t \cup \Gamma_0$, where Γ_u is the part of the boundary with prescribed displacements, Γ_t is the part of the boundary with prescribed traction forces, and Γ_0 is the part of the boundary with traction-free boundary conditions. Consider the linearized elasticity system

$$\begin{cases} -\nabla\sigma(u(x)) &= p(x) \quad \text{in } \Omega \\ u(x) &= \bar{u} \quad \text{in } \Gamma_u \\ \sigma(u(x))\cdot n &= \bar{t}(x) \quad \text{in } \Gamma_t \\ \sigma(u(x))\cdot n &= 0 \quad \text{in } \Gamma_0 \end{cases}, \tag{1}$$

where u is the displacement field, σ is the Cauchy stress tensor, p and $\bar{t}$ are the body and surface forces, $\bar{u}$ is the prescribed displacement field, and n is the unit outward normal vector to $\partial\Omega$. The stress tensor σ and the symmetric gradient of the displacement field ε are related by the constitutive equation

$$\mathbb{C}_{ijkl}(x)\varepsilon_{ij}(u(x)) = \sigma_{kl}(u(x)), \tag{2}$$

where $\mathbb{C}_{ijkl}$ represents the fourth order constitutive tensor. Considering the mechanical design as a body occupying a domain Ω^m which is part of the reference domain Ω, we split the design into two subdomains with the following characteristic function

$$\Theta(x) = \begin{cases} 1 & x \in \Omega^m \\ 0 & x \in \Omega \setminus \Omega^m \end{cases}. \tag{3}$$

We can formulate the topology optimization problem as the minimization of the structural compliance over admissible design and displacement fields satisfying equilibrium equation in its weak form as follows

$$\begin{aligned} \min_{u\in U,\Theta} \quad & J(\Theta) = \textstyle\int_\Omega p(x)u(x)\ d\Omega + \int_{\Gamma_t} \bar{t}(x)u(x)\ d\Gamma \\ \text{s. t.} : \quad & a(u(x),v(x)) = l(v(x)) \quad \forall\ v \in \mathcal{V} \\ : \quad & \mathbb{C}_{ijkl}(x) = \Theta(x)\ \mathbb{C}^0_{ijkl} \\ : \quad & Vol(\Omega^m) = \textstyle\int_\Omega \Theta(x) d\Omega \leqslant V^*, \end{aligned} \tag{4}$$

where $\mathcal{V}$ denotes the space of kinematically admissible displacement fields, $\mathbb{C}^0_{ijkl}$ is the stiffness tensor for an elastic material, V^* is the volume target considering the pointwise volume fraction $\Theta(x)$ for a black-and-white design, and $a(\cdot,\cdot)$ and $l(\cdot)$ are the bilinear and linear forms, respectively, as follows

$$\begin{aligned} a(u(x),v(x)) &= \textstyle\int_\Omega \sigma(u(x))\varepsilon(v(x)) d\Omega \\ &= \textstyle\int_\Omega \mathbb{C}_{ijkl}(x)\varepsilon_{ij}(u(x))\varepsilon_{kl}(v(x)) d\Omega \\ l(v(x)) &= \displaystyle\int_\Omega p(x)v(x)\ d\Omega + \int_{\Gamma_t} \bar{t}(x)v(x)\ d\Gamma. \end{aligned} \tag{5}$$

The SIMP method relaxes this integer-based problem introducing an interpolation scheme that penalizes a continuous density variable ρ, $0 \leqslant \rho \leqslant 1$, characterizing composite materials [50] and allowing us to use gradient-based approaches in the optimization. The following power-law interpolation function permits us to rewrite the constitutive tensor equation as

$$\mathbb{C}_{ijkl}(\rho(x)) = \mathbb{C}^{min}_{ijkl} + \rho(x)^p\left(\mathbb{C}^0_{ijkl} - \mathbb{C}^{min}_{ijkl}\right), \tag{6}$$

where p > 1 is the penalization power, and $\mathbb{C}^0_{ijkl}$ and $\mathbb{C}^{min}_{ijkl}$ are the fouth order constitutive tensors of the stiff and soft material respectively. This penalization function relates the variable $\rho(x)$ and the material tensor $\mathbb{C}_{ijkl}$ in the equilibrium analysis, satisfying $\mathbb{C}_{ijkl}(0) = \mathbb{C}^{min}_{ijkl}$ and $\mathbb{C}_{ijkl}(1) = \mathbb{C}^0_{ijkl}$. We can select p sufficiently big to penalize intermediate densities. According to Bendsøe and Sigmund [50], we usually require p $\geqslant$3 to ensure that the Hashin–Shtrikman bounds are not violated. The introduction of the weak phase $\mathbb{C}^{min}_{ijkl}$ mimicking void material allows us to avoid the singularity of the rigidity matrix during the optimization [14]. The problem can be then stated as

$$\begin{aligned} \min_{u\in U,\rho(x)} \quad & J(u(\rho(x))) = \textstyle\int_\Omega p(x)u(x)\ d\Omega + \int_{\Gamma_N} \bar{t}(x)u(x)\ d\Gamma \\ \text{s. t.} : \quad & a(u(x),v(x)) = l(v(x)) \quad \forall\ v \in \mathcal{V} \\ : \quad & \mathbb{C}_{ijkl}(\rho(x)) = \mathbb{C}^{min}_{ijkl} + \rho(x)^p\left(\mathbb{C}^0_{ijkl} - \mathbb{C}^{min}_{ijkl}\right) \\ : \quad & \textstyle\int_\Omega \rho(x) d\Omega \leqslant V^*,\ \rho(x) \in [0,1], \end{aligned} \tag{7}$$

where $\rho(x)$ is the design variable ranging from solid ($\rho = 1$) to void ($\rho = 0$). This relative density field $\rho(x)$ is constant within each finite element, being the design variables the relative densities of the finite elements. We denote the design domain by Ω and constrain the volume of material $V(\boldsymbol{\rho})$ to be smaller than a prescribed target V^*. However, density-based topology optimization is prone to numerical instabilities due to checker-board patterns appearing as penalizing intermediate material densities, mesh-dependency as refining the tessellation of the continuum, and the presence of local minima in the design space [51]. We can introduce a density measure $\tilde{\rho}(x) = \rho(x)$ that tends to regularize the problem addressing these numerical instabilities.

Besides, we use projection techniques $\bar{\tilde{\rho}}(x) = \tilde{\rho}(x)$ to project the filtered designs into solid/void space, which produce designs with a clear physical interpretation [52]. The problem can be then discretized using the Galerkin finite element method as follows

$$\begin{aligned} \min_{\boldsymbol{\rho}} \quad & f\left(\bar{\tilde{\boldsymbol{\rho}}}\right) = \mathbf{F}^{\mathrm{T}}\mathbf{U} = \mathbf{U}^{\mathrm{T}}\mathbf{K}\mathbf{U} = \sum_e \mathbf{u}_\mathbf{e}^{\mathrm{T}}\mathbf{k}_\mathbf{e}\mathbf{u}_\mathbf{e} \\ \text{s. t.}: \quad & \mathbf{K}\mathbf{U} = \mathbf{F} \\ : \quad & E(\bar{\tilde{\rho}}_e) = E_{min} + \bar{\tilde{\rho}}_e^{p}(E_0 - E_{min}) \\ : \quad & \frac{\sum_e \bar{\tilde{\rho}}_e v_e}{V_T} - V^* \leqslant 0, \;\; \rho(x) \in [0,1], \end{aligned} \tag{8}$$

where f is the objective function, $\mathbf{K}$ is the global stiffness matrix, $\mathbf{U}$ and $\mathbf{F}$ are the displacement and force vectors respectively, $E(\bar{\tilde{\rho}}_e)$ is the artificial Young's modulus of an element, $\mathbf{E}_0$ and $\mathbf{E}_{min} > 0$ are the Young's modulus of solid and void material respectively, and V_T is the total volume without material penalization. Lower-case variables represent the element-wise quantities; v_e is the volume of element e and $k_e = E(\bar{\tilde{\rho}}_e)\mathbf{k}_\mathbf{e}^\mathbf{0}$ is the element stiffness matrix, being $\mathbf{k}_\mathbf{e}^\mathbf{0}$ the element stiffness matrix with $\mathbf{E}_0$ Young's modulus. We use the density measure introduced in [53], the so-called density filter, to regularize the density field $\boldsymbol{\rho}$ as follows

$$\tilde{\rho}_e = \frac{\sum_{i\in N_e} w(\mathbf{x}_i) v_i \rho_i}{\sum_{i\in N_e} w(\mathbf{x}_i) v_i} \tag{9}$$

where $\tilde{\boldsymbol{\rho}}$ is the filtered elemental density field, N_e is the neighborhood set of elements lying within the radius R, and $w(\cdot)$ is the weighting function $w(\mathbf{x}_\mathbf{i}) = R - |\mathbf{x}_\mathbf{i} - x_e|$. This filtering technique allows us to cope with numerical problems in topology optimization, in particular, checkerboard patterns and mesh-dependent designs. The existence of the solution using such a filter was mathematically proven by Bourdin [53].

We use the parametrized projection function suggested by Xu et al. [52], which projects the filtered density values $\tilde{\boldsymbol{\rho}}$ above a threshold η to solid and the values below to void using the following smooth function

$$\bar{\tilde{\rho}}_e = \begin{cases} \eta\left[e^{-\beta\left(1-\frac{\tilde{\rho}_e}{\eta}\right)} - \left(1-\frac{\tilde{\rho}_e}{\eta}\right)e^{-\beta}\right] & 0 \leqslant \tilde{\rho}_e \leqslant \eta \\ (1-\eta)\left[1 - e^{-\beta\frac{(\tilde{\rho}_e-\eta)}{1-\eta}} + \frac{\tilde{\rho}_e-\eta}{1-\eta}e^{-\beta}\right] + \eta & \eta \leqslant \tilde{\rho}_e \leqslant 1 \end{cases}, \tag{10}$$

where the projection parameter β allows us to control the smooth function. By using the threshold value $\eta = 0$, we obtain a similar projected field than using the Heaviside step filter introduced by Guest et al. [54], which ensures a minimum length scale on the solid phase. The threshold value $\eta = 1$ performs similar filtering to the modified Heaviside filter introduced by Sigmund [55], giving rise to a minimum length scale on the void phase. The expression (10) can be written as

$$\bar{\tilde{\rho}}_e = \frac{\tanh(\beta\eta) + \tanh(\beta(\tilde{\rho}_e - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))}, \tag{11}$$

which provides an efficient alternative to calculate the projection avoiding the contitional statements [56].

We can derive the sensitivity of the objective function (8) to the design variable $\rho(x)$ using the chain rule as

$$\frac{\partial f(\bar{\tilde{\rho}})}{\partial \rho_i} = \frac{\partial f(\bar{\tilde{\rho}})}{\partial \bar{\tilde{\rho}}_e}\frac{\partial \bar{\tilde{\rho}}_e}{\partial \tilde{\rho}_e}\frac{\partial \tilde{\rho}_e}{\partial \rho_i}, \tag{12}$$

obtaining the different terms are as follows

$$\frac{\partial f(\bar{\tilde{\rho}})}{\partial \bar{\tilde{\rho}}_e} = -\mathbf{u}^{*\mathrm{T}}\frac{\partial \mathbf{K}}{\partial \rho}\mathbf{u} = -\mathbf{u}^{\mathrm{T}} p \bar{\tilde{\rho}}_e^{p-1}(E_0 - E_{min})\mathbf{k}_\mathbf{e}^\mathbf{0}\mathbf{u}, \tag{13}$$

$$\frac{\partial \bar{\tilde{\rho}}_e}{\partial \tilde{\rho}_e} = \frac{\beta(\mathrm{sech}(\beta(\tilde{\rho}_e - \eta)))^2}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))} \tag{14}$$

$$\frac{\partial \tilde{\rho}_e}{\partial \rho_i} = \frac{w(\mathbf{x}_i) v_i}{\sum_{i\in N_e} w(\mathbf{x}_i) v_i} \tag{15}$$

where $\mathbf{u}^*$ is given by the solution of the adjoint problem $\mathbf{K}\mathbf{u}^* = \frac{\partial f}{\partial \mathbf{u}} = f$, whose solution is $\mathbf{u}^* = \mathbf{u}$ because the minimization of the structural compliance is self-adjoint.

We update the design variables using the Method of Moving Asymptotes (MMA) proposed by Svanberg [57] for its excellent parallel scalability [58]. The MMA method is suitable for addressing inequality-constrained optimization problems, such as the formulation of (8), by generating and solving a series of approximate convex subproblems instead of the original non-linear problem. The algorithm stops when we reach the maximum number of iterations or when the change variable $||\rho_{e_{k+1}} - \rho_{e_k}||$ and the change in the objective function $||f_{k+1} - f_k||$ fall below a prescribed value.

3. Adaptive mesh refinement and coarsening

Let G be an element containing *vertices*, *edges*, and *faces* in three-dimensional cases. We define a mesh $\{G_i\}_{i=1}^{N_e}$ as the union of the N_e elements such that $\Omega = \cup_i G_i$ is a connected and bounded region. Let denote boundary elements as ∂G and interior elements as $\langle G \rangle$. Assuming that interior elements do not intersect, i.e. $\langle G_i \rangle \cap \langle G_j \rangle = \varnothing \; \forall \; i \neq j$, we state that a mesh is *conforming* if the set $\partial G_i \cap \partial G_j \; \forall \; i \neq j$ is a shared vertex, shared edge, shared face, or an empty set. We define a *hanging node* as a vertex lying in the interior of an edge, or a face in three-dimensional cases. We call *non-conforming* meshes the ones containing at least one *hanging node*. The refinement of one *parent* element G_i consists of replacing it with at least two *child* elements G_{ik} such that $G_i = \cup_k G_{ik}$. For example, we divide the edges of a parent 2D rectangle into two edges, creating a new vertex in the center of the parent rectangle and one vertex in the center of each parent edge. We use these vertices to create the four rectangles composing the refined *child* elements. We follow a similar approach for 3D bricks, creating a new vertex in the center of the 3D brick and dividing each face into four faces. We use them to create the eight *child* brick elements from the parent brick element. We remove parent elements after refining them, storing the links to the child elements in a *refinement tree*. This information facilitates the coarsening by reintroducing the parent element and removing the child elements. We say that a non-conforming mesh is consistent if lower-dimensional mesh entities (vertices, edges, and faces) are either identical or a proper subset of the other. We refer to the smaller entity as a slave and the larger ones containing other entities as a master, whereas lower-dimensional mesh entities that are neither masters nor slaves are called conforming entities.

Most AMR methods constrain the hanging nodes using algebraic operators expressing the constrained degrees of freedom (DOFs) as a linear combination of the unconstrained DOFs. We use the approach proposed by Çervenyét al. [49] for non-conforming AMR of unstructured meshes. This method cast the elimination of constrained DOFs as a form of variational restriction decoupling the AMR method and the governing equation. We can discretize the weak variational problem of (7) constructing an approximate finite-dimensional subspace $\mathcal{V}_h \subset \mathcal{V}$ on the $\{G_i\}_{i=1}^{N_e}$ mesh assembling the stiffness matrix $\mathbf{K}$ and vector of load forces $\mathbf{F}$ such that

$$\mathbf{V_h^T K U_h} = \mathbf{V_h^T F} \quad \mathbf{V_h} \in \mathbb{R}^d, \tag{16}$$

where d is the dimension of the problem. This is equivalent to the linear system of Eqs. (8), whose solution provides an approximate solution of vector $\mathbf{U_h}$. In non-conforming meshes, we have a larger partially conforming space $\widehat{\mathcal{V}}_h \supset \mathcal{V}_h$ with $\widehat{\mathcal{V}}_h \in \mathbb{R}^{\widehat{d}}$ and $\widehat{d} > d$ since slave entities containing hanging nodes have DOFs that are independent of master DOFs. We can restore the conformity in the interfaces containing hanging nodes constraining slave entities (slave DOFs) by interpolating the finite element functions of their masters. We obtain the solution vector $\widehat{\mathbf{U}_h}$ in the partially conforming space $\widehat{\mathcal{V}}_h$ as

$$\widehat{\mathbf{U}}_h = \begin{pmatrix} \mathbf{U_h} \\ \mathbf{W_h} \end{pmatrix} \quad \mathbf{U_h} \in \mathbb{R}^d, \ \mathbf{W_h} \in \mathbb{R}^{\widehat{d}-d}, \tag{17}$$

where $\mathbf{W_h}$ represents all slave DOFs which can be evaluated by the linear interpolation $\mathbf{W_h = W\,U_h}$ using the interpolation operator $\mathbf{W}$. We can write (17) as

$$\widehat{\mathbf{U}_h} = \mathbf{P_c U_h}, \quad \text{with } \mathbf{P_c} = \begin{pmatrix} \mathbf{I} \\ \mathbf{W} \end{pmatrix}, \tag{18}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{P_c}$ is the conforming prolongation matrix. We can assemble the stiffness matrix $\widehat{\mathbf{K}}$ and load vector $\widehat{\mathbf{F}}$ in the space $\widehat{\mathcal{V}}_h$ obviating the hanging nodes in the mesh. The corresponding linear system $\widehat{\mathbf{K}}\widehat{\mathbf{U}_h} = \widehat{\mathbf{F}}$ results in a non-conforming solution where the slave DOFs are not constrained. However, it taking the expression (18) the variational formulation on $\mathbf{V_h}$ becomes

$$\mathbf{V_h^T P_c^T \widehat{K} P_c U_h} = \mathbf{V_h^T P_c^T \widehat{F}} \quad \mathbf{V_h} \in \mathbb{R}^d. \tag{19}$$

Thus, we can solve the smaller problem $\mathbf{P_c^T \widehat{K} P_c U_h} = \mathbf{P_c^T \widehat{F}}$ where the slave DOFs are eliminated and then prolonging $\mathbf{U_h}$ to obtain the conforming solution $\widehat{\mathbf{U}_h} \in \widehat{\mathcal{V}}_h$. We remark that factoring $\mathbf{P_c}$ out of the AMR governing equation facilitates the incorporation of AMR in different problems. The challenge is then the efficient construction of this $\mathbf{P_c}$ operator, which consists of constructing the identity $\mathbf{I}$ and the slave interpolation $\mathbf{W}$ submatrices of $\mathbf{P_c}$. We refer to the work of Červený et al. [49] and the implementation of the AMR method using the mfem [59] and hypre [60] libraries for the details of the serial and parallel construction of the $\mathbf{P_c}$ operator.

The AMR technique for coarsening and h-refinement requires an error estimator to determine the areas of interest. Error estimators based on design variables usually aim to locate the material boundary of the optimum topology, such as refinement indicators based on the continuity of material to find elements dominated by solid or void material regions [61]. These refinement indicators identify the design variables all along the apparent edge of the structure where we require a higher accuracy to calculate the sensitivity of the objective function, which guides the topology optimization during the optimization process. Besides, the computation of regions with weak material contributes significantly to the overall computational cost but little to the accuracy of the optimized design. Thus, error estimators identifying sets of design variables with void material for coarsening can increase the performance meaningfully [35]. We use and evaluate the following error estimators

$$\varepsilon_v = \frac{\int_{\Omega_{rt}} \tilde{\bar{\rho}}(x) d\Omega_{rt}}{\int_{\Omega_{rt}} d\Omega_{rt}}, \tag{20}$$

$$\varepsilon_g = \frac{\partial f(\tilde{\bar{\rho}}(x))}{\partial \rho_i}, \tag{21}$$

where ε_v is the error estimator based on the design variable, ε_g is the error estimator based on the sensitivity of the objective function, and $\Omega_{rt} \subset \Omega$ is the part of the domain in the different levels of the *refinement tree*: the parent element for refinement and the set of child elements for coarsening to the parent element. We can choose the threshold $\varepsilon_v \leqslant E_{min} + \varepsilon$, being ε a small value to detect the void elements for coarsening. However, there exists a relationship between the norm of maximum relative density gradient and element size [62]. Thus, the thresholding using the gradient-based criteria requires the heuristic tunning to provide similar results to the one obtained using a fine mesh.

4. Parallel strategy

The partition of complex models into smaller and more manageable pieces is a common approach to use distributed computational resources. This strategy allows us to divide the computational burden and the memory requirements across the computational resources of the distributed computational system. The underline idea consists of partitioning the domain into a set of subdomains and then solving these subdomains in parallel. This strategy increases the overall performance by distributing computing and memory requirements. We call the techniques using this strategy Domain Decomposition Methods (DDMs). The performance of these methods depends on the workload for solving the subdomains and the volume of communications. Efficient partitioning techniques allow us to optimize the former factor, whereas the variants of DDMs aim to optimize the latter. We adopt a simple Global Subdomain Implementation (GSI) [63] as solving strategy.

Fig. 1 shows the flowchart of the distributed memory implementation of density-based topology optimization using h-refinement and coarsening. We divide the problem into several subdomains to calculate the recursive stages of the topology optimization method. After the partitioning, we initialize the communications needed to perform the distributed operations: the AMR operations, the filtering of design variables, and the solving of the system of equations using FEA. It is well-known that FEA is the principal bottleneck of the topology optimization pipeline, and thus we use the coarsening to reduce the computational cost of this stage. This operation requires the projection of the design variables to penalize the Young modulus of finite elements. We assemble and solve the distributed system of equations using the coarse mesh. Since the analysis for evaluating the objective function is the bottleneck of the processing, the assembly and resolution in the coarse mesh can increase the performance of the whole process meaningfully. We achieve this improvement in the case the design domain is highly void, which usually occurs after some iterations of the optimization. We then refine the coarse mesh by projecting the resulting displacement field, performing the other stages of topology optimization using the fine mesh. Since we calculate the sensitivities and update the design variables in the fine mesh, the evolution of the optimization is similar to obviating the AMR technique allowing to arise members even in coarsened areas of the coarse mesh, which are not coarsen again if the error estimator is not satisfied after the sensitivity calculation and density update. Next, we present the details of the parallel implementation of all the topology optimization stages.

4.1. Domain partitioning

The formulation of density-based topology optimization allows us to use the same partitioning during the whole optimization process. This fact is a crucial point to achieving an efficient implementation. We tessellate the initial domain into several non-overlapping subdomains in the initialization using such subdomains in the stages of the optimization loop shown in Fig. 1. The partitioning algorithm takes the resulting mesh of the

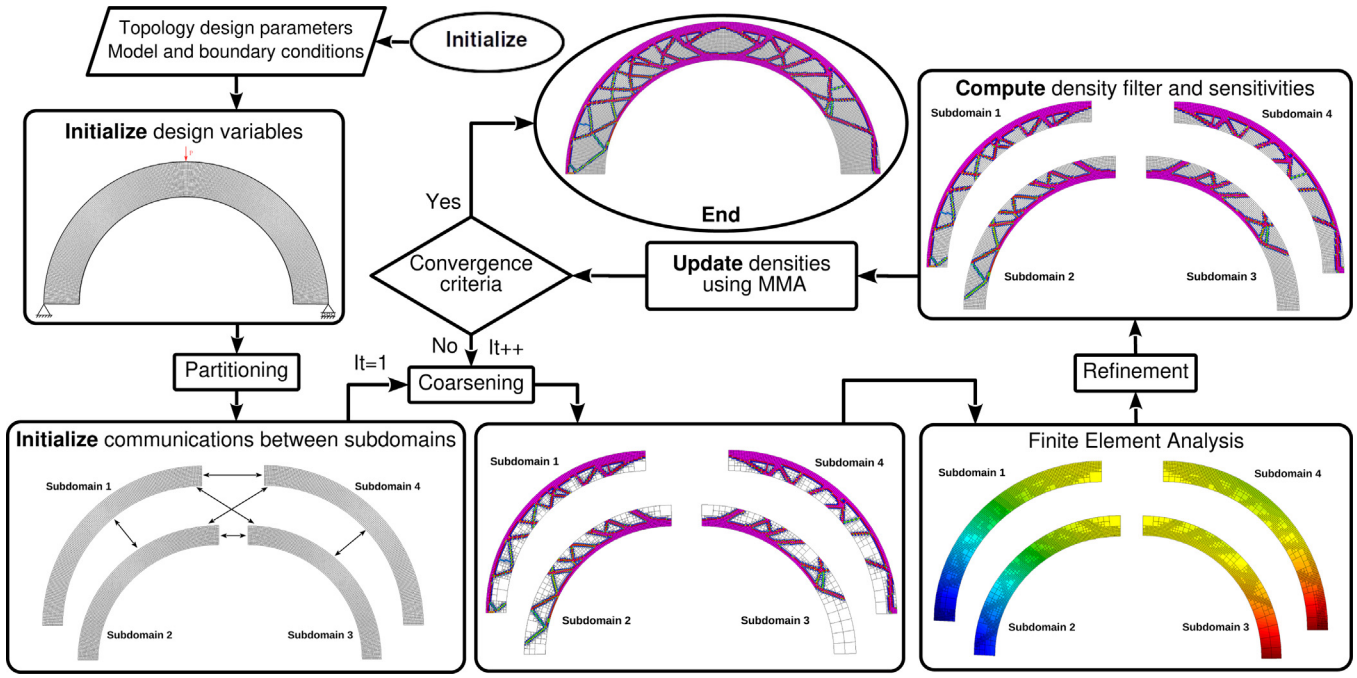

Fig. 1. Flowchart of the distributed density-based topology optimization using h-refinement and coarsening.

geometric discretization to divide it following optimization criteria. In particular, the minimization of the number of interface elements reduces the data exchange between processes making the computation associated with each part.

We generate a dual graph from the mesh of the finite element model to perform the partitioning. The vertices of such a graph are the mesh elements, and the arcs are the connections between them. We divide the graph nodes using a multilevel k-way method [64] to define the subdomains considering optimization criteria. These optimization criteria include the minimization of the resulting subdomain connectivity graph and the contiguous partition enforcement. The efficiency of the partition approach is of paramount importance since the partition method is memory-intensive, which is particularly true for large-scale problems and for partitioning with a high number of subdomains. We use ParMETIS [65] library to perform this parallel partitioning using the MPI standard.

4.2. Distributed solving

Let consider the conforming linear system of equations to solve the problem stated in (19) as

$$\mathbf{A}\mathbf{U}_\mathbf{h} = \mathbf{B}, \tag{22}$$

where $\mathbf{A} = \mathbf{P}_\mathbf{c}^\mathbf{T}\widehat{\mathbf{K}}\mathbf{P}_\mathbf{c} \in \mathbb{R}^{n_u \times n_u}$ is the coeffient matrix, $\mathbf{B} = \mathbf{P}_\mathbf{c}^\mathbf{T}\widehat{\mathbf{F}} \in \mathbb{R}^{n_u \times 1}$ is the right-hand side vector, $\mathbf{U}_\mathbf{h} \in \mathbb{R}^{n_u \times 1}$ is the solution vector, and n_u is the number of unknowns. We require a distributed representation of the coefficient matrix and vectors to evaluate the objective function of expression (8) using multi-core architectures. Let us assume that the coefficient matrix $\mathbf{A}$ using compressed sparse row (CSR) format is distributed across $p = \{1, \ldots, n_p\}$ processes, with n_p the number of computing processes, by contiguous blocks of rows as follows

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}^0 \\ \vdots \\ \mathbf{A}^{p-1} \end{pmatrix}, \tag{23}$$

where the computation of each block submatrix $\mathbf{A}^{p-1} \in \mathbb{R}^{n^{p-1} \times n_u}$ is performed by one single processor p with n^{p-1} the number of rows of the block submatrix $\mathbf{A}^{p-1}$ and n_u the number of columns of $\mathbf{A}$. The block submatrices use the global row indices $\{\mathbf{n^0}, \ldots, \mathbf{n^{p-1}}\}$ to facilitate the operations between subdomains. We store these submatrices into local $\mathbf{A}_{loc}^{p-1}$ and remote $\mathbf{A}_{rem}^{p-1}$ parts using the hypre library [60]. The former is a square matrix with the "local" n^{p-1} unknowns, whereas the latter is a matrix containing coefficients with global column indices stored in other processors. This data structure allows us to differentiate between "local" and "distributed" computation. Local computation is performed with the data stored in the own process p, whereas distributed computation requires some communication mechanism. We use a similar approach for the distributed dense vector $\mathbf{B} \in \mathbb{R}^{n^{p-1} \times 1}$ using the global row indices $\{\mathbf{n^0}, \ldots, \mathbf{n^{p-1}}\}$.

The communications make use of the standardized and portable MPI mechanism. In the initialization, the processes p calculate the global columns required from both the own and other processes. Data exchange consists of receiving ghost values from the processes sharing unknowns (global column indices) and then sending the data to the processes that require them to form the gathered vectors. We perform data exchange directly between computing processes since each processor p knows its receive and send processors. This data exchange procedure reduces the computational complexity and the storage requirements because the number of neighbors and the amount of data are independent of the number of p processors.

We use a distributed Krylov subspace method preconditioned with a multigrid method for solving the linear system of equations of (22); in particular, a distributed conjugate gradient iterative solver. Structural mechanics problems commonly use multigrid methods as a preconditioner of Krylov subspace methods for solving. The underline idea is that the interpolation operator of the multigrid approach will hardly be optimal, which makes it less efficient for some specific error components. The convergence is slow when this occurs despite almost all of the error components being reduced quickly. A Krylov subspace method for eliminating these

error components is usually a more efficient solution than improving the construction of the interpolation operator.

Multigrid methods use a two-grid scheme to address the problem. The central idea of these methods is that the "smooth error" $\mathbf{e}$ that is not eliminated by relaxation should be removed by coarse-grid correction. We remove this "smooth error" by solving the residual $\mathbf{r}$ as $\mathbf{Ae} = \mathbf{r}$ on a coarser grid and then interpolating the error back to the fine grid correcting the fine-grid approximation by $\mathbf{U_h} \leftarrow \mathbf{U_h} + \mathbf{e}$. Let l be the grid level and n_l the number of unknowns in that l level. Considering the initialization $\mathbf{A}_0 = \mathbf{A}$ and $n_0 = n$, we define the coarse grid coefficient matrices $\mathbf{A}_{l+1}$ recursively as follows

$$\mathbf{A}_{l+1} = \mathbf{R}_l \mathbf{A}_l \mathbf{P}_l, \tag{24}$$

where $\mathbf{P}_l \in \mathbb{R}^{n_l \times n_{l+1}}$ is the interpolation or prolongation operator, $\mathbf{R}_l \in \mathbb{R}^{n_{l+1} \times n_l}$ (normally obtained as $\mathbf{P}_l^T$) is the restriction operator, and $\mathbf{A}_l \in \mathbb{R}^{n_l \times n_l}$ and $\mathbf{A}_{l+1} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$ are the fine and coarse grid coefficient matrices, respectively. We calculate recursively such transfer operators until a grid level L in which the number of unknowns n_L is sufficiently low to solve it in a reasonable time.

There are two basic multigrid approaches [66]: GMG and AMG. GMG uses the geometry of the problem to define the grid coefficient matrices and the transfer operators. We refer the readers to [67] for the details of a parallel implementation of GMG methods. On the other hand, AMG methods only use the information available in the linear system of equations, allowing their use as a "black-box" function in finite element codes. Multigrid approaches have two stages: setup and solving. The former aims to find the proper transfer operators, whereas we compute the latter in all the iterations of the recursive solver. The construction of the transfer operators is hard to parallelize, whereas the solving consists of matrix–vector products, which are easily computed in parallel. In this work, we use an AMG as a preconditioner of the distributed conjugate gradient solver to evaluate the objective function of topology optimization. There exist different choices for coarsening, interpolation, and smoothing in the implementation. We use the BoomerAMG [68,69] because it provides parallel coarsening and interpolation functionalities for the classical AMG method [70].

Algorithm 1.

Algorithm 1: AMG setup

Input: $\mathbf{A}, \theta, \tau, n_u$
Output: $\mathbf{A_1}, \mathbf{P_0}, \mathbf{R_0}, \ldots, \mathbf{A_{n_L}}, \mathbf{P_{n_L-1}}, \mathbf{R_{n_L-1}}$

$\mathbf{A_l} \leftarrow \mathbf{A}$, $l \leftarrow \emptyset$, $n_l \leftarrow n_u$;
while n_l is too big to solve **do**
  $\mathbf{C_l}, \mathbf{F_l} \leftarrow part(\mathbf{A_l}, \theta)$; // Partition into disjoint sets
  $\mathbf{\Omega_{l+1}} \leftarrow \mathbf{C_l}$; // Initialize grid at level $l+1$
  $\mathbf{P_l} \leftarrow interp(\mathbf{\Omega_{l+1}}, \mathbf{F_l}, \tau)$; // Construct interpolation
  $\mathbf{R_l} \leftarrow \mathbf{P_l^T}$; // Construct restriction
  $\mathbf{S_l} \leftarrow Smoother(\mathbf{A_1})$; // Setup smoother
  $\mathbf{A_{l+1}} \leftarrow \mathbf{R_l A_l P_l}$; // Galerkin projection
  $l \leftarrow l+1$, $n_l \leftarrow n_{l+1}$;
end

Algorithm 1 describes the AMG setup for the classical AMG method. The setup stage requires the coefficient matrix $A \in \mathbb{R}^{n_u \times n_u}$, a strength threshold θ for the coarsening, and a truncation factor τ for constructing the interpolation operator. The AMG setup generates the coarsening grid Ω_{l+1} by separating the coefficients of the system of equations into either C-coefficients, grouped into $\mathbf{C_l}$ and taken to the next level, and F-coefficients, grouped into $\mathbf{F_l}$ and interpolated by the C-coefficients. The AMG setup introduces a strong dependence criterion calibrated by the strength threshold θ [69]. We use the Hybrid Modified Independent Set (HMIS) algorithm for parallel coarsening, which is obtained by combining the Parallel Modified Independent Set (PMIS) algorithm [71] with a one-pass Ruge-Stüben scheme. After partitioning the coefficient matrix $\mathbf{A_l}$ at the corresponding level l into the disjoint sets $\mathbf{C_l}$ and $\mathbf{F_l}$, we construct the interpolation or prolongation $\mathbf{P_l}$ operator determining the influence of F-coefficients to the C-coefficients by the interpolation of weights. We remove the coefficients of the interpolation operator smaller than a truncation factor τ to avoid an excessive number of non-zero coefficients. We then construct the restriction operator from the interpolation operator. We also generate a smoother operator to reduce the oscillatory error components. In particular, we use the parallel implementation of sparse approximate inverse preconditioner ParaSails [72], which uses a priori sparsity patterns and least-squares minimization. Finally, we obtain the coefficient matrix at the next level $\mathbf{A_{l+1}}$ performing the distributed sparse Galerkin projection. By using these algorithms, we can calculate the AMG setup stage in parallel.

Algorithm 2.

Algorithm 2: AMG solving (V-cycle)

Input: $\mathbf{r_l}$, $\mathbf{s_l}$, l, $\mathbf{A_1}$, $\mathbf{S_1}$, μ_1, μ_2
Output: $\mathbf{s_l}$

// Pre-relaxation: smooth μ_1 times on $\mathbf{S_l}$ $\mathbf{s_l} = \mathbf{r_l}$
$\mathbf{s_l} \leftarrow \text{Smooth}(\mathbf{r_l}, \mathbf{S_l}, \mathbf{s_l}, \mu_1)$;
$\mathbf{r_l} \leftarrow \mathbf{r_l} - \mathbf{A_l}\,\mathbf{s_l}$; // Compute residual
$\mathbf{r_{l+1}} \leftarrow \mathbf{R_l}\,\mathbf{r_l}$; // Restict residual to coarse grid
// Coarsest (last level L) grid
if $(l+1 == L)$ **then**
  $\mathbf{s_{l+1}} \leftarrow \mathbf{A_{l+1}}^{-1}\mathbf{r_{l+1}}$; // Solve
else
  $\mathbf{s_{l+1}} \leftarrow \mathbf{R_l}\,\mathbf{s_l}$; // Restict solution to coarse grid
  // Recursion
  $\mathbf{s_{l+1}} \leftarrow \text{V-cycle}(\mathbf{r_{l+1}}, \mathbf{s_{l+1}}, l+1, \mathbf{A_{l+1}}, \mathbf{S_{l+1}}, \mu_1, \mu_2)$;
  $\mathbf{s_l} \leftarrow \mathbf{P_l}\,\mathbf{s_{l+1}}$; // Prolongation
  // Post-relaxation: smooth μ_2 times on $\mathbf{S_l}$ $\mathbf{s_l} = \mathbf{r_l}$
  $\mathbf{s_l} \leftarrow \text{Smooth}(\mathbf{r_l}, \mathbf{S_l}, \mathbf{s_l}, \mu_2)$;
end

Algorithm 2 shows the pseudo-code of the V-cycle of classical AMG for preconditioning the distributed conjugate gradient iterative solver. This algorithm aims to approximate the solution of (22) given the residual of the previous estimation of the iterative solver. The procedure consists of applying μ_1 smoothing operations to the approximate solution s_l at the level l and the computation of the residual r_l for the relaxed approximate solution s_l. We then restrict the residual to the coarse grid and solve the linear system if we reach the last level L. We prolongate the solution s_l at the coarsest grid to the finer one applying μ_2 smoothing operations to the approximate solution.

Algorithm 3 details the pseudo-code of the distributed conjugate gradient algorithm using the V-cycle of classical AMG as a preconditioner. This iterative solver requires the maximum number of iterations $\max_{\text{iter}}$, the tolerance $< \text{tol}, \text{tol}_{\text{abs}} >$, the coefficient matrix $\mathbf{A}$, the right-hand side $\mathbf{B}$, and the initial guess $\mathbf{U_h^0}$ for initial-

izing the iterative procedure. The recursive solver provides an approximate solution $\mathbf{U_h}$ of (22) with a residual after *it* iterations.

Algorithm 3.

Algorithm 3: Conjugate gradient algorithm

Input: $\max_{\text{iter}}$, tol, tol_{abs}, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U_h^0}$, μ_1, μ_2
Output: $\mathbf{U_h}$, $^{\|\mathbf{r}\|_2}/_{f_{norm}}$, it

$it \leftarrow \emptyset$, $\gamma \leftarrow \emptyset$, $\mathbf{s} \leftarrow \emptyset$, $\mathbf{p} \leftarrow \emptyset$, $\mathbf{q} \leftarrow \emptyset$, $\mathbf{U_h} \leftarrow \mathbf{U_h^0}$;
$f_{norm} \leftarrow \|\mathbf{B}\|_2$;
$\mathbf{r} \leftarrow \mathbf{B}$ - $\mathbf{A}$ $\mathbf{U_h}$;
$\varepsilon \leftarrow \max(\text{tol} \cdot \|\mathbf{r}\|_2, \text{tol}_{\text{abs}})$;
while $(it < max_{iter})$ && $(\|\mathbf{r}\|_2 > \varepsilon)$ **do**
  // AMG V-cycle to estimate $\mathbf{A}^{-1}\mathbf{r}$
  $\mathbf{s} \leftarrow$ V-cycle$(\mathbf{r}, \mathbf{s}, \emptyset, \mu_1, \mu_2)$;
  $\gamma_{\text{old}} \leftarrow \gamma$, $\gamma \leftarrow \mathbf{r}^T\mathbf{s}$;
  if $(it == \emptyset)$ **then**
    $\mathbf{s} \leftarrow \mathbf{p}$;
  else
    $\mathbf{p} \leftarrow \mathbf{s} + {}^{\gamma}/_{\gamma_{\text{old}}}\mathbf{p}$; // axpby operation
  end
  $\mathbf{q} \leftarrow \mathbf{A}\,\mathbf{p}$; // spmv operation
  $\alpha \leftarrow {}^{\gamma}/_{\mathbf{p}^T\mathbf{q}}$;
  $\mathbf{U_h} \leftarrow \mathbf{U_h} + \alpha\,\mathbf{p}$; // axpby operation
  $\mathbf{r} \leftarrow \mathbf{r}$ - $\alpha\,\mathbf{q}$; // axpby operation
  $it \leftarrow it + 1$;
end

The distributed conjugate gradient algorithm using the V-cycle of classical AMG as a preconditioner allows us to obtain the solution of (19) using parallel computing. We then have to apply (18) to obtain the solution of the FEA using non-conforming meshes in parallel.

4.3. Distributed filtering

Density filtering replaces the dependence of the material properties on its pointwise density regularizing the density field using the mean of a convolution operator. We require the design variables of the elements surrounding each cell within the radius R defined in (9) to calculate such a convolution operation or conic filter. By using AMR in the density field, we have to consider the values of the design variables in the different layers of the *refinement tree* of neighbor elements [62]. However, the search of the neighbor elements in different layers has a high computational cost, which is significantly incremented in distributed memory parallel implementations. Since we perform the filtering in the fine mesh, the search of the neighbor elements in the different layers is not needed. Nevertheless, the parallel implementation of the filtering in distributed memory systems using non-overlapped DDM methods requires communications for sharing design variables between subdomains.

Fig. 2 shows the problem for elements close to the border of the subdomain. We can observe that the communications required by each design variable depend on the domain partitioning, and we only need to share information between close subdomains. The full circles in the center of finite elements represent the design variables to regularize. In the detail of two design variables, we depict the local and remote design variables using gray and empty circles, respectively, needed to perform the convolution operation. We also

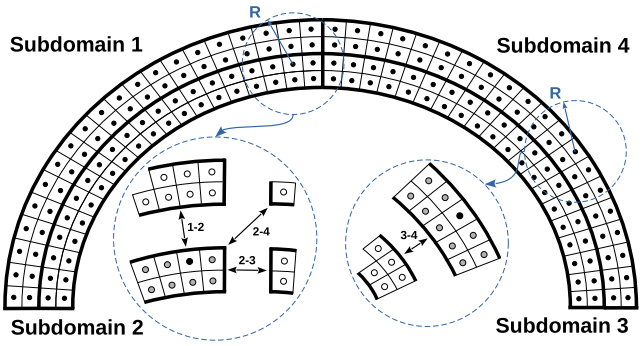

Fig. 2. Density filtering with communication between subdomains.

show the communications required between subdomains to compute (9). Since the performance of this operation is of paramount importance to achieve an efficient implementation of density-based topology optimization, we search the information about the neighbor elements and the communications needed by each design variable as a preprocessing step.

The preprocessing creates hash tables storing for each design variable the following information to calculate the convolution operation: the element indexes of the local design variables within the radius R, the element and subdomain indexes of remote design variables within the radius R, and the Euclidean distance from the design variable to local and remote design variables within the radius R. This information allows us to use point-to-point communications between the subdomains sharing data, minimizing the sharing information in problems with a high amount of subdomains, which facilitates scalability.

Thus, the preprocessing increases the computational performance at the cost of a higher memory requirement to store the previously mentioned hash tables. We have to remark that the construction of the hash tables has high computational requirements when we use large R values. Nevertheless, we largely compensate the preprocessing stage during the topology optimization iterations for relatively small values of the radius R. Alternatively, we can regularize the density field implicitly as the solution of a Helmholtz type partial differential equation (PDE) with homogeneous Neumann boundary conditions [73]. We can solve this PDE in parallel using the techniques presented in the previous section using a prescribed tolerance for the iterative solver. In our experience, a PDE for filtering the density field is suitable for large values of the radius R and using simplices due to the higher information to share for the regularization. We also have to remark that using a large radius R for the regularization can modify the conditioning of the linear system of equations, and thus the results presented in this work can be different.

4.4. Distributed optimization and sensitivity calculation

Once we divide the domain into several non-overlapped subdomains, the parallel computation of the sensitivities (12) is straightforward because the different terms using the chain rule only require the information of the design variable (14), the result from the FEA (13), and the result from the filtering (15). We follow the parallel implementation of MMA proposed by Aage and Lazarov [58], implementing a continuation strategy for the β parameter of the projection (11) following a heuristic strategy.

5. Numerical Experiments

We evaluate the benefits and limitations of the proposed parallel implementation of density-based topology optimization using

AMR schemes to reduce the computational bottleneck of the optimization, the FEA for evaluating the objective function. We test the performance of preconditioning using AMG and GMG multigrid methods during the optimization using different radii for regularizing. We also check the improvement in performance using AMR strategies and the strong scaling in two- and three-dimensional topology optimization problems. We present the cumulative wall-clock time for the whole topology optimization process to show the computing benefits while providing similar results than obviating the use of AMR techniques.

We run the numerical experiments using two computation systems: one workstation and two workstations connected with a 10 Gigabit Ethernet network working as a cluster. The former allows us to evaluate the performance using a workstation, whereas the latter shows the scalability increasing the distributed computational resources. We equip the former workstation using an AMD Ryzen™ 9 3950X CPU with 16 cores (32 simultaneous multithreading) at 3.5 GHz (turbo core speed of 4.7 GHz) and 96 GB of RAM, whereas we use two Intel Xeon E5-2687 W v4 CPUs with 12 cores (24 hyper-threading) at 3.0 GHz (turbo boost speed of 3.5 GHz) per CPU connected with dual LGA2011-v3 and 256 GB of RAM per computing node in the cluster. Thus, the computing nodes can run 24 processes in parallel.

The battery of experiments consists of four topology optimization problems using structured meshes with finite elements using different aspect ratios to show that the approach is not limited to any constraint, such as computational patterns. In particular, the experiments consist of the topology optimization of a Messerschmitt-Bölkow-Blohm (MBB) aircraft floor beam, a cantilever beam, a curved beam, and a dome. The first experiment is a two-dimensional plane stress problem, whereas the other experiments are three-dimensional models. The MBB and cantilever beam experiments tesselate the domain with elements using a similar aspect ratio, whereas the curve beam and dome experiments discretize the domain adapting to its geometry.

We parameterize the geometry and meshes to facilitate reproducible results. Table 1 specifies the geometric and topology optimization parameters of the experiments. All the experiments use a tolerance $tol = 10^{-6}$ for the distributed conjugate gradient method presented in Algorithm (3 and a penalization power $p = 3$ for the power-law interpolation function of (8). We set Young's modulus of solid $E_0 = 1$ and void $E_{min} = 10^{-9}$ material in the power-law interpolation function of (8). The numerical experiments using the AMG preconditioner use a strength threshold $\theta = 0.5$ for the coarsening, a truncation factor $\tau = 0.0$ (no truncation) for constructing the interpolation operator, and an HMIS algorithm for parallel coarsening. The numerical experiments using GMG preconditioning use a damping factor w_s for the Jacobi smoothing that we calibrate to ensure convergence in the different experiments [32]. Both multigrid approaches use the same number of pre and post-smoothing steps; in particular, $\mu_1 = \mu_2 = 4$. We follow a continuation strategy for the β parameter of the projection (11) following a heuristic strategy for each experiment. We also reuse the displacement solution of the previous iteration of topology optimization because it improves the convergence of the solving of the system of equations in topology optimization [33].

We solve the experiments using multi-core computing with distributed memory using the standardized and portable MPI mechanisms. We evaluate the performance of the preconditioner of the distributed iterative solver using AMG and GMG methods for two- and three-dimensional problems, showing the advantages of each one in different types of experiments. We calculate the regularization, sensitivities, and optimization using the fine mesh, whereas we use the dynamic coarse mesh for the analysis. This strategy requires adaptive coarsening and refinement up to the fine mesh at all topology optimization iterations. We evaluate the computational cost of the dynamic AMR strategy, checking the increment in performance of the whole process with the reference implementation without using AMR. We present below the numerical results of the four density-based topology optimization experiments evaluating the computational aspects of the parallel AMR strategy using multi-core computation.

5.1. MBB beam experiment

The first experiment consists of the topology optimization of a two-dimensional MBB beam with fixed displacements on both bottom parts of endpoints and a vertical load at the top center of the beam. Fig. 3(a) shows the geometry and boundary conditions of the finite element model, indicating the geometric parameters specified in Table 1. This table also includes the topology optimization design parameters. In particular, the volume fraction v and the element size e_{s1} used for regularizing the design field. Fig. 3(b) shows the symmetry simplification used for the topology optimization, including the boundary conditions. Fig. 3(c) depicts a coarse mesh partitioned into eight subdomains using the ParMetis library. We can observe that the partitioning is performed into balanced subdomains enforcing the contiguous partitions. It also shows the parameterization of the tessellation with the *div* variable. We use the number of refinement levels l_{ref} to obtain the dynamic coarsening mesh for the FEA.

Fig. 4 depicts the final design of the MBB experiment with $div = 512$ elements using different radii R for the regularization. In particular, we show the final design using two, four, six, and eight times the element size e_{s1} specified in Table 1. This tessellation generates a model with 786432 elements and 1576962 unknowns. We can observe that by reducing the size used for the regularization, we can capture more details in the final design. We initialize the projection parameters $\eta = 0.5$ and $\beta = 1.0$ of (11) following a heuristic strategy duplicating the value of β every

Table 1
The geometric and topology design parameters of experiments.

Geometry (meters)			Topology parameters		
MBB beam					
	L (m)	**H** (m)	**v (%)**	$\mathbf{e_{s1}}$ (m)	
	6.0	1.0	15	0.0019531	
Cantilever beam					
W (m)	**L** (m)	**H** (m)	**v (%)**	$\mathbf{e_{s1}}$ (m)	$\mathbf{e_{s2}}$ (m)
1.0	2.0	1.0	5	0.015625	0.0078125
Curved beam					
W (m)	**R** (m)	**H** (m)	**v (%)**	$\mathbf{e_{s1}}$ (m)	$\mathbf{e_{s2}}$ (m)
1.0	4.0	1.0	3	0.015708	0.007854
Dome					
	$\mathbf{R_{ext}}$ (m)	$\mathbf{R_{int}}$ (m)	**v (%)**	$\mathbf{e_{s1}}$ (m)	$\mathbf{e_{s2}}$ (m)
	1.0	0.96	4	0.005	0.0025

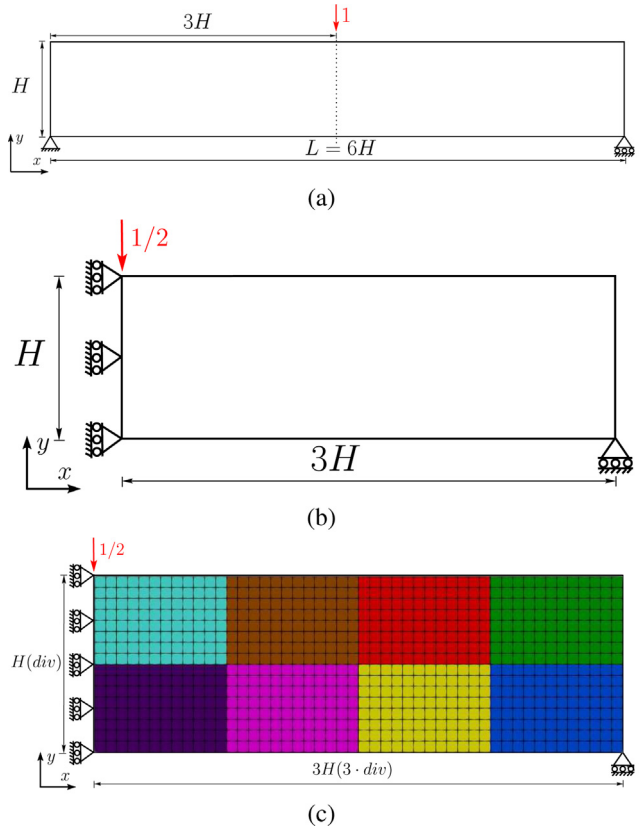

Fig. 3. The MBB experiment: (a) geometric configuration, (b) symmetry simplification, and (c) mesh parameterization and partitioning into eight subdomains.

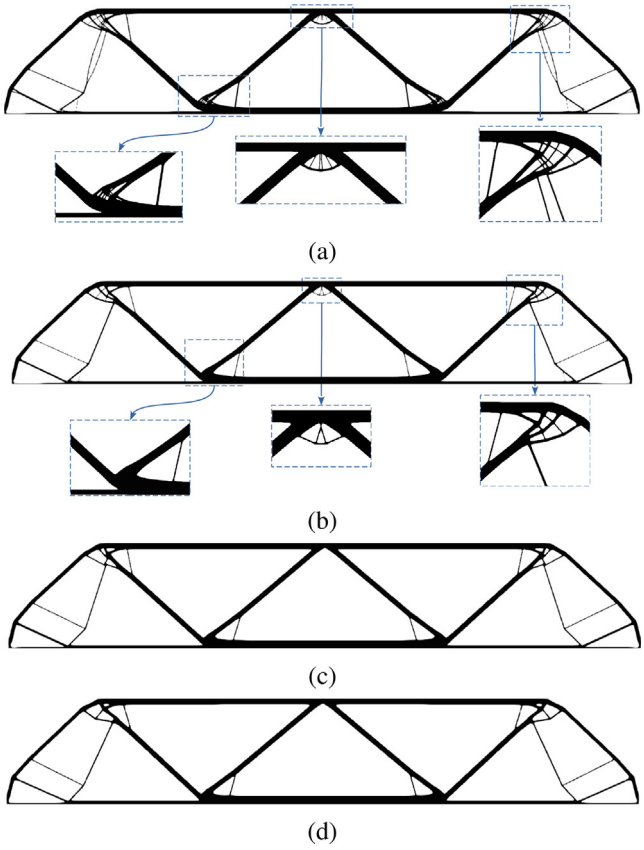

Fig. 4. The MBB experiment with 1536 × 512 elements of edge size e_{s1} regularizing with (a) $R = 2e_{s1}$, (b) $R = 4e_{s1}$, (c) $R = 6e_{s1}$, and (d) $R = 8e_{s1}$.

15 iterations from iteration 60 until the $\beta = 64$ value. We evaluate the computational performance preconditioning the distributed conjugate gradient using AMG and GMG multigrid methods with the different regularization sizes of R for (9). We tune the damping factor $w_s = 0.75$ for the Jacobi smoothing to ensure convergence using GMG preconditioning. We do not use AMR in these experiments, and they are the reference methods to evaluate the performance increment using the proposal. The GMG method uses six levels in the V-Cycle of Algorithm 2 from a grid of 1536 × 512 elements to a mesh of 24 × 8, whereas the AMG method uses twelve levels from a sparse coefficient matrix of 1576962 × 1576962 coefficient elements to a coefficient matrix of 8 × 8 unknowns. Fig. 5 shows the total wall-clock time for solving the MBB experiment using AMG and GMG preconditioning with distributed memory computing using 16 computing processes. This wall-clock time includes the setup of the multigrid preconditioners and solving stages. We can observe that the convergence of the iterative solver depends on the regularization size using GMG preconditioning, whereas it is similar using the AMG preconditioning approach. The GMG preconditioning provides better results increasing the regularization size R, and it is less efficient than the preconditioning using AMG for this two-dimensional experiment.

By choosing the regularization size $R = 2e_{s1}$ because it provides more details about the optimum distribution of material, we evaluate the use of the proposal using AMR to improve the efficiency of analysis. We use six refinement levels $l_{ref} = 6$ from the grid of 1536 × 512 elements for dynamic coarsening constructing the *refinement tree*. Fig. 6(e) shows the details of the coarsening corresponding to the design shown in Fig. 4(a). We can observe the six levels of child elements with the corresponding hanging nodes in the details of the two blue dashed bounding boxes. We evaluate the proposal using two error estimators: the error estimator ε_v of

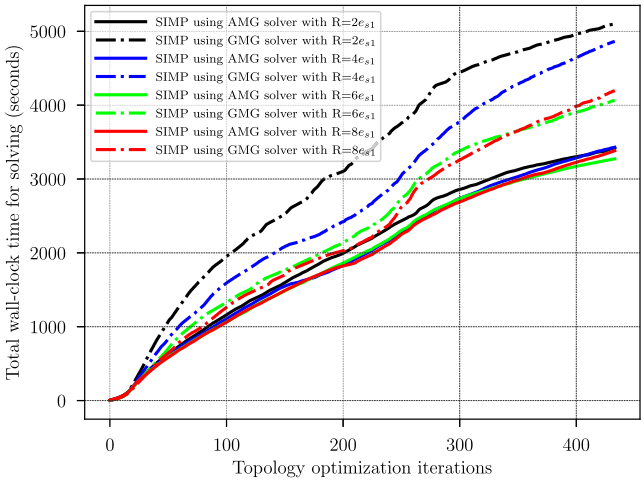

Fig. 5. The MBB experiment with 1536 × 512 elements of edge size e_{s1}: cumulative wall-clock time for solving with 16 cores on one host including setup and solving stages of AMG and GMG multigrid methods using different regularization sizes of R for (9).

(20) based on the value of the design variable and ε_g of (21) based on the value of the gradient of compliance. We use the thresholds $\varepsilon_v \leqslant 2E_{min}$ and $\varepsilon_g \leqslant 10^{-10}$ as criteria for dynamic coarsening the mesh. The coarsening using the thresholding approach based on the value of the design variables is called void-based strategy since it aims to locate areas with weak material, whereas the coarsening using the heuristic thresholding using the gradient of the objective function is called gradient-based approach.

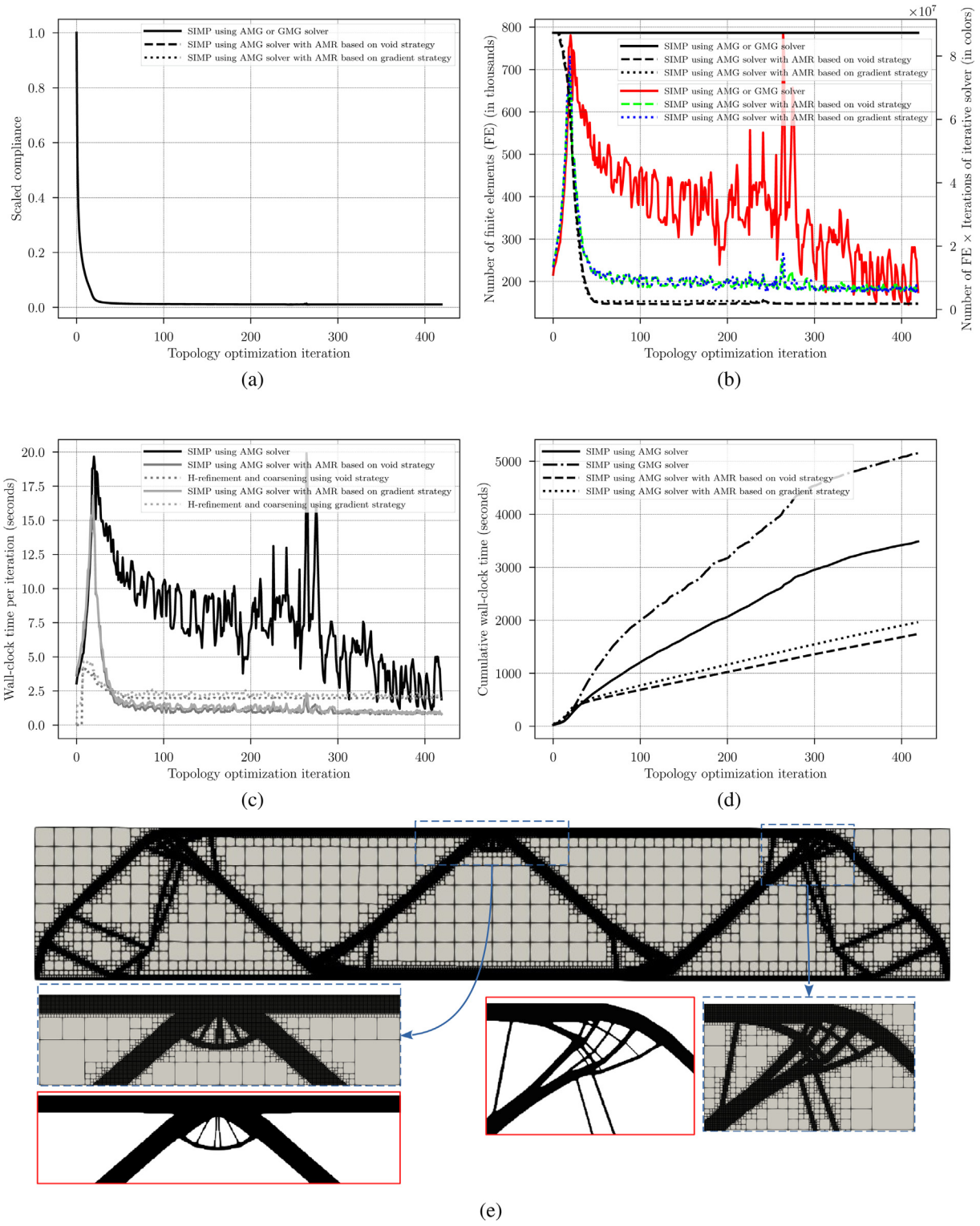

Fig. 6. The MBB experiment with 1536 × 512 elements using h-refinement and coarsening with gradient- and void-based strategies: (a) the compliance evolution, (b) the number of finite elements used for solving and the metric for estimating the computational cost, (c) the wall-clock time per iteration including solving stages and dynamic refinement using 16 cores on one host, (d) the total wall-clock time for the optimization using 16 cores on one host, and (e) the refinement detail of the design variables.

We show the evolution of compliance during the optimization using different approaches in Fig. 6(a). We can observe that the objective function evolution using GMG and AMG preconditioning combined with dynamic AMR with different error estimators is similar. Fig. 6(b) depicts the number of finite elements used during the optimization process (black color) and a metric (in colors) of the computational cost: the product of the number of finite elements and the number of iterations needed by the iterative solver to reach the prescribed tolerance. The initial iteration of topology optimization with AMR approaches uses the same amount of finite elements as the optimization without AMR. The number of finite elements using AMR techniques quickly decreases until the

optimization capture the shape of the design, and then it maintains a stable number of finite elements to extract the details. We obtain a significant improvement in the metric of the computational cost by reducing the number of finite elements. Nevertheless, it is similar again in the last iterations of the topology optimization. In our experience, we cannot reduce the threshold value ε_g of the gradient-based approach because it suffers from numerical instabilities due to the continuation strategy for the β parameter of the projection contribution of (14) to the gradient of the objective function using the chain rule following the heuristic strategy.

We evaluate the wall-clock time per iteration for solving the MBB experiment using AMR and the size $R = 2e_{s1}$ for regularizing. Fig. 6(c) shows the wall-clock time per iteration, including the setup and solving stages and the dynamic coarsening and refinement up to six refinement levels $l_{ref} = 6$ for FEA using 16 cores on the workstation with an AMD Ryzen™ 9 3950X CPU. We can observe that the wall-clock time per iteration without using AMR reduces as the optimization progresses due to the reuse of the displacement solution of the previous iteration of topology optimization [33]. The wall-clock time for solving using AMR strategies reduces significantly after some iterations of the topology optimization. However, the distributed coarsening and refinement operations have a non-negligible computational cost, which is particularly true using a relevant number of refinement levels $l_{ref} = 6$ and when the design is close to the solution because the problem is better conditioned. Thus, we have to find a trade-off between the cost of solving and the computational requirements to reduce the system of equations to solve.

Fig. 6(d) shows the cumulative wall-clock time for all the processing of MBB experiment with regularization size $R = 2e_{s1}$ and using 16 computing cores in one workstation, including the distributed FEA, coarsening and refinement using six refinement levels $l_{ref} = 6$, regularization, sensitivity calculation, and optimization. We can observe that the increment in computational cost is negligible to the wall-clock time for solving shown in Fig. 5. Thus, the FEA dominates the computational cost of the whole process using a distributed multi-core implementation. We also can observe that the cumulative wall-clock time is similar using AMR and without using it in the initial iterations of the topology optimization, whereas we reduce this computing time significantly as the topology optimization progresses. We can obtain a higher speedup of the whole process by increasing the iterations performed in the topology optimization. The distributed implementation using AMR void-based strategy provides better results than the one using the gradient-based approach. We attribute this fact to the numerical problems related to the continuation strategy of β for the projection, mainly due to we cannot adjust the threshold of the error estimator ε_g based on the gradient of compliance for instabilities in the calculation of the contribution to the sensitivity of the objective function of (14) when doubling the β parameter.

Fig. 7 shows the wall-clock time percentage for the different stages of MBB topology optimization with 1536 × 512 elements using 16 cores on one host. We depict these percentages using concentric doughnut charts, showing from the outer ring to the inner one the results using AMG preconditioning, GMG preconditioning, AMR with the void-based strategy using AMG preconditioning, and AMR with the gradient-based approach using AMG preconditioning. We can observe a significant reduction of the percentage of the wall-clock time for the assembly stage using AMR techniques because we assemble the system of equations on the coarse mesh. We can notice an increment in the percentage of the wall-clock time for calculating the sensitivities and for the design variable update using MMA because we perform these tasks on the fine mesh. Finally, we can observe that the percentage of wall-clock time for coarsening and refinement using AMR techniques is relevant for two-dimensional problems. Coarsening and refinement stages take more than 60% of the wall-clock time. The solving stage takes the 40% of the wall-clock time approximately. For this reason, it is of paramount importance to find a trade-off between the cost of solving and the computational requirements to reduce the system of equations to solve. We can do this by tunning the maximum number of levels for the coarsening and refinement of the mesh using AMR techniques.

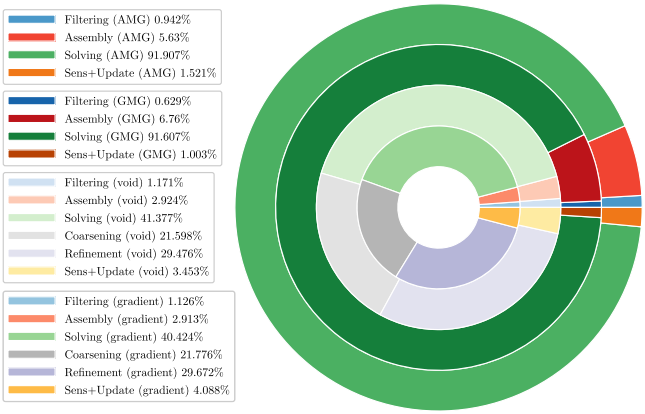

Fig. 7. Percentage of wall-clock time for the different stages of MBB topology optimization with 1536 × 512 elements using 16 cores on one host.

We evaluate the strong scaling for the MBB experiment using different computational resources. Fig. 8 shows the cumulative wall-clock time for the topology optimization with regularization size $R = 2e_{s1}$ using different amounts of computational resources: one, four, eight, and sixteen computing cores on one workstation. Fig. 8(a) shows the strong scaling without using AMR. We can observe that the wall-clock time decreases as increasing the computational resources preconditioning with GMG and AMG. We obtain a speedup of more than 5x using sixteen computing cores with the reference approach using one computing core. The speedup is similar using GMG and AMG preconditioning. Fig. 8(b) shows the cumulative wall-clock time using AMR with the resources previously mentioned. We can observe that the wall-clock time also decreases as increasing the computational resources, obtaining an additional speedup of more than 2x for this experiment. We have to remark that these speedups are for topology optimization, obtaining higher accelerations for the last topology optimization iterations.

5.2. Cantilever beam experiment

The cantilever experiment consists of the topology optimization of a three-dimensional beam with fixed displacements on one side and loads uniformly distributed along the lower edge of the other side. Fig. 9(a) shows the geometry and boundary conditions of the finite element model, indicating the geometric parameters specified in Table 1. This table also includes the topology optimization design parameters: the volume fraction v and the element size e_{s1} and e_{s2} used for regularizing the design field of the experiments using one workstation and two computing nodes. Fig. 4(b) depicts a coarse mesh partitioned into eight subdomains using the ParMetis library for enforcing contiguous partitioning with balanced subdomains. It also shows the parameterization of the tessellation with the *div* variable, that we coarse dynamically up to the refinement level l_{ref} for increasing the performance of the analysis.

Fig. 10 shows the results of the cantilever experiment with $div = 64$ using a radius of $R = 2e_{s1}$, with e_{s1} specified in Table 1, to regularize the density field. This tessellation generates a model

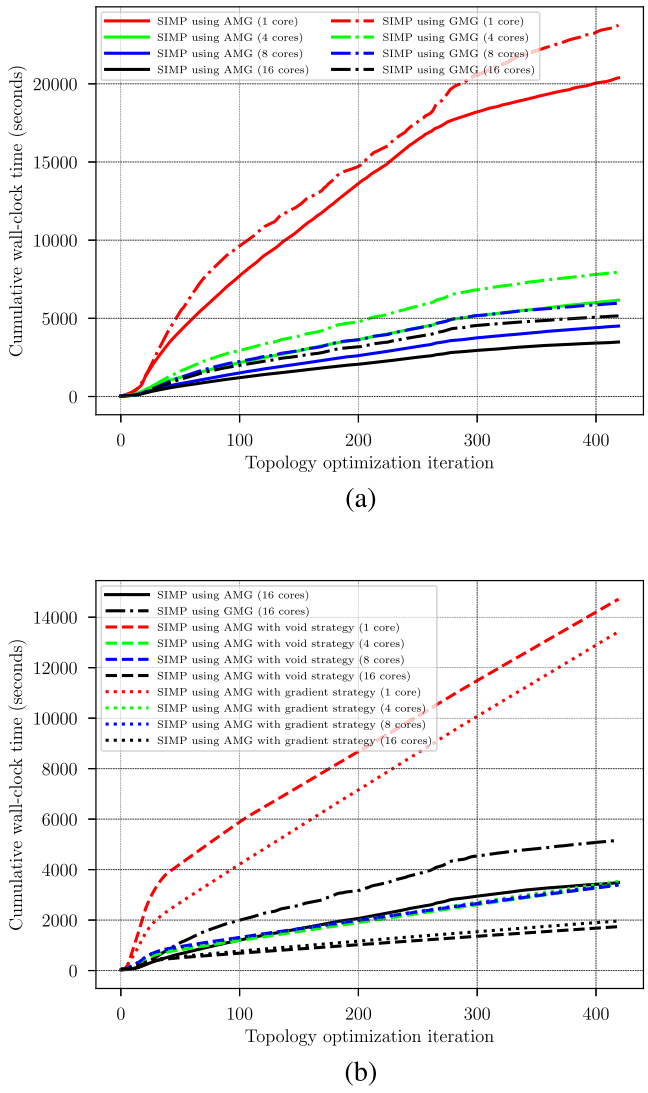

Fig. 8. Strong scaling for the MBB experiment with 1536 × 512 elements of edge size e_{s1} regularizing with $R = 2e_{s1}$: total wall-clock time (a) using AMG and GMG multigrid solvers, and (b) using AMG multigrid solver with h-refinement and coarsening based on gradient and void strategies.

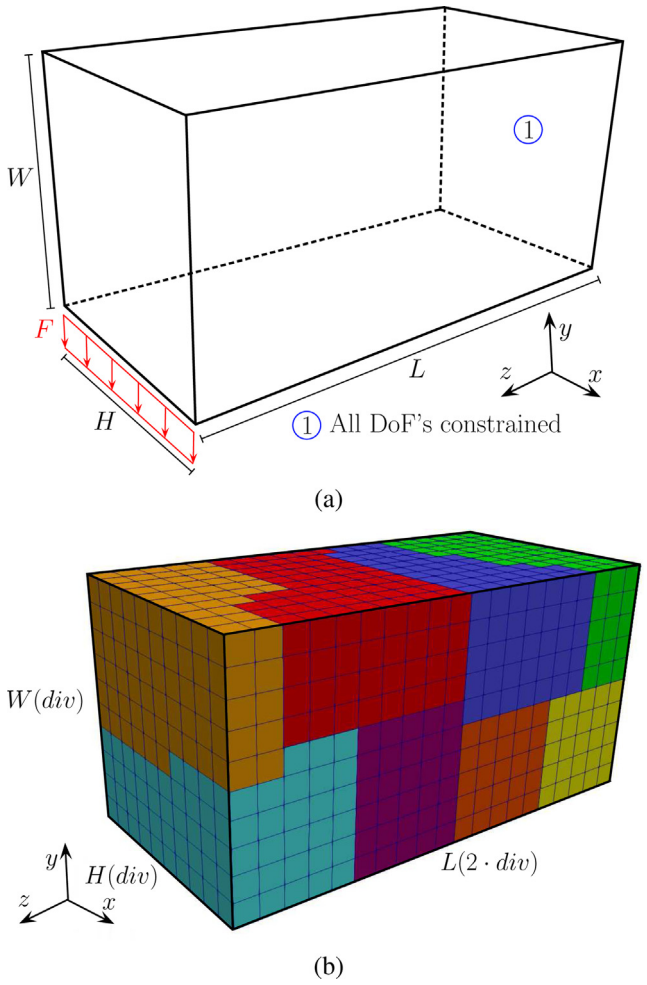

Fig. 9. The cantilever experiment: (a) geometric configuration and boundary conditions, and (b) mesh parameterization and partitioning into eight subdomains.

with 524288 finite elements and 1635075 unknowns. We use one workstation for all the calculations. We initialize the projection parameters $\eta = 0.3$ and $\beta = 1.0$ of (11) following a heuristic strategy duplicating the value of β every 15 iterations from the iteration 60 until the $\beta = 64$ value. We tune the damping factor $w_s = 0.45$ for the Jacobi smoothing to ensure convergence using GMG preconditioning. The GMG method uses four levels in the V-Cycle of Algorithm 2 from a grid of 64 × 64 × 128 elements to a mesh of 4 × 4 × 8, whereas the AMG method uses nine levels to prolongate and restrict from a sparse coefficient matrix of 1635075 × 1635075 coefficient elements to a coefficient matrix of 9 × 9 unknowns. We run the experiments using one workstation with an AMD Ryzen™ 9 3950X CPU. We use four refinement levels $l_{ref} = 4$ from the grid of 64 × 64 × 128 elements for dynamic coarsening constructing the *refinement tree*. Fig. 10(e) shows the details of the coarsening corresponding to the final design of the cantilever experiment, where we can observe the four levels of child elements with the corresponding hanging nodes. We use the thresholds $\varepsilon_v \leqslant 2E_{min}$ and $\varepsilon_g \leqslant 10^{-7}$ as criteria for dynamic coarsening the mesh using void-based and gradient-based approaches.

Fig. 10(a) depicts the evolution of compliance during the topology optimization using different approaches with and without AMR. We can observe that the objective function evolution using GMG and AMG preconditioning and AMG preconditioning combined with dynamic AMR using many error estimators is similar. Fig. 10(b) depicts the number of finite elements used during the topology optimization process (black color) and a metric (in colors) of the computational cost: the product of the number of finite elements and the number of iterations needed by the iterative solver to reach the prescribed tolerance. The initial iterations of topology optimization with AMR approaches use the same amount of finite elements as the optimization without AMR. We can observe a significant reduction of the number of finite elements, almost ten times less, using a smaller number of refinement levels $l_{ref} = 4$ than the two-dimensional MBB experiment. The number of finite elements using AMR techniques quickly decreases until the optimization capture the shape of the design, and then it maintains a stable number of finite elements to extract the details. We obtain a significant improvement in the metric of the computational cost by reducing the number of finite elements. In this three-dimensional case, we can observe this improvement during the whole topology optimization. In contrast to the two-dimensional MBB experiment, we do not notice numerical instabilities due to the continuation strategy for the β parameter of the projection contribution of (14) to the gradient of the objective function using the chain rule following the heuristic strategy. Nevertheless, we use a

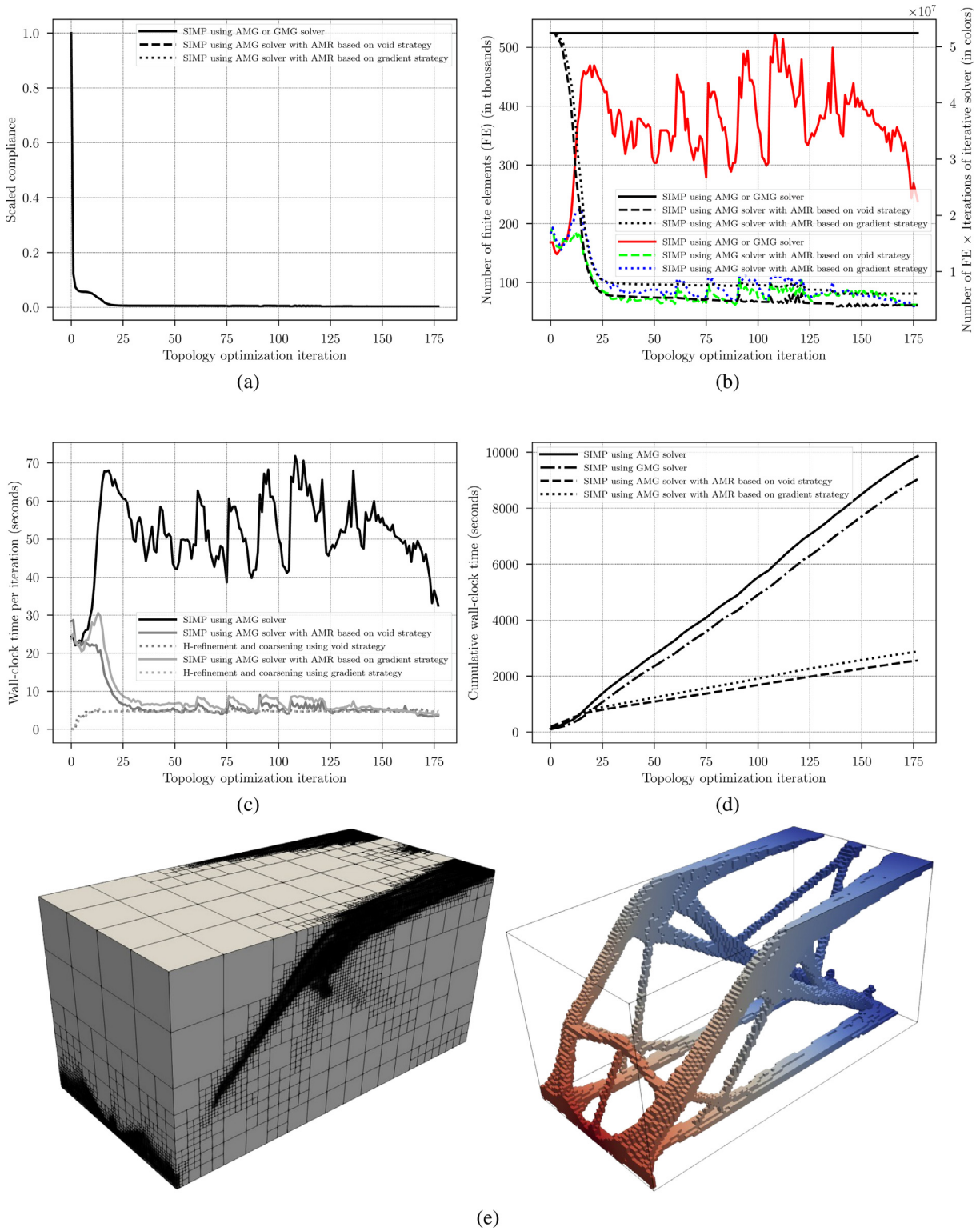

Fig. 10. The cantilever experiment with 64 × 64 × 128 elements of edge size e_{s1} regularizing with $R = 2e_{s1}$ and using h-refinement and coarsening with gradient- and void-based strategies: (a) compliance evolution, (b) the number of finite elements used for solving and the metric for estimating the computational cost, (c) the wall-clock time per iteration including solving stages and dynamic refinement using 16 cores on one host, (d) the total wall-clock time for the optimization using 16 cores on one host, and (e) the refinement detail of design variables and the final design showing the displacement field.

conservative threshold value for the error estimator ε_g based on the gradient of the objective function.

Fig. 10(c) shows the wall-clock time per iteration for solving the cantilever experiment with and without AMR using the size $R = 2e_{s1}$ to regularize the density field. The timing includes the setup and solving stages and the dynamic coarsening and refinement up to four refinement levels $l_{ref} = 4$ for FEA using 16 cores on the workstation with an AMD Ryzen™ 9 3950X CPU. We

observe that the wall-clock time per iteration without using AMR reduces in the last iterations of the optimization progress by reusing the displacement solution of the previous iteration of topology optimization. We also observe a significant reduction of the wall-clock time for solving using AMR strategies. The wall-clock time for computing the distributed coarsening and refinement operations increases as the number of finite elements for analysis reduces. However, the computational cost of AMR is more than compensated with the wall-clock time saved in the solving.

Fig. 10(d) shows the cumulative wall-clock time for all the processing of the cantilever experiment with regularization size $R = 2e_{s1}$ and using 16 computing cores in one workstation including the distributed analysis, coarsening, and refinement using four refinement levels $l_{ref} = 4$, regularization, sensitivity calculation, and design variable update. We can observe that the solver using GMG preconditioning is slightly faster than the resolution using the AMG preconditioner. However, we obtain a more than relevant speedup using AMR approaches. In particular, we achieve an acceleration of more than 3x for the topology optimization using 175 iterations using as reference the topology optimization using GMG preconditioning. Nevertheless, we obtain higher speedups when we coarse the mesh for only capturing the details of the shape of the final design.

Fig. 11 shows the wall-clock time percentage for the different stages of the three-dimensional topology optimization with 64 × 64 × 128 elements using 16 cores on one host. We depict the information using the same format as Fig. 7. We can observe a relevant reduction of the percentage of the wall-clock time for the assembly stage using AMR techniques because we assemble the system of equations on the coarse mesh. We notice that the increment in the percentage of the wall-clock time for calculating the sensitivities and for the design variable update using MMA is significantly higher than the two-dimensional experiment shown in Fig. 7. We attribute this to a higher acceleration of the three-dimensional topology optimization problems. We also can observe that the percentage of wall-clock time for coarsening and refinement using AMR techniques in this three-dimensional problem is lower than the two-dimensional topology optimization presented above. We attribute this to the lower number of levels for coarsening and refinement than the two-dimensional topology optimization problems.

We evaluate the strong scaling for the cantilever experiment using different computational resources. Fig. 12 shows the cumulative wall-clock time for the topology optimization of the cantilever model of 64 × 64 × 128 elements with regularization size $R = 2e_{s1}$ using a different number of computing cores: one, four, eight, and sixteen computing cores on one workstation. Fig. 12(a) shows the

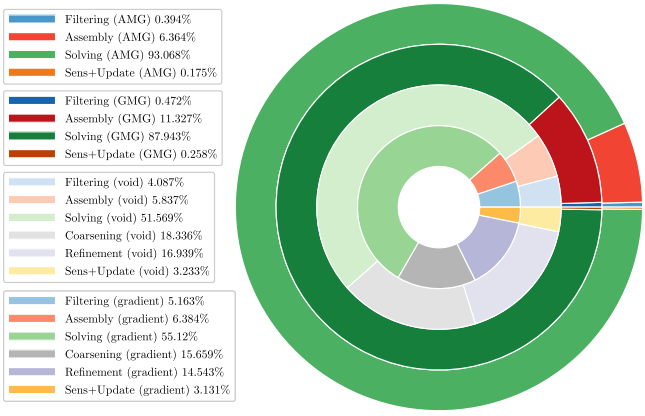

Fig. 11. Percentage of wall-clock time for the different stages of the cantilever topology optimization with 64 × 64 × 128 elements using 16 cores on one host.

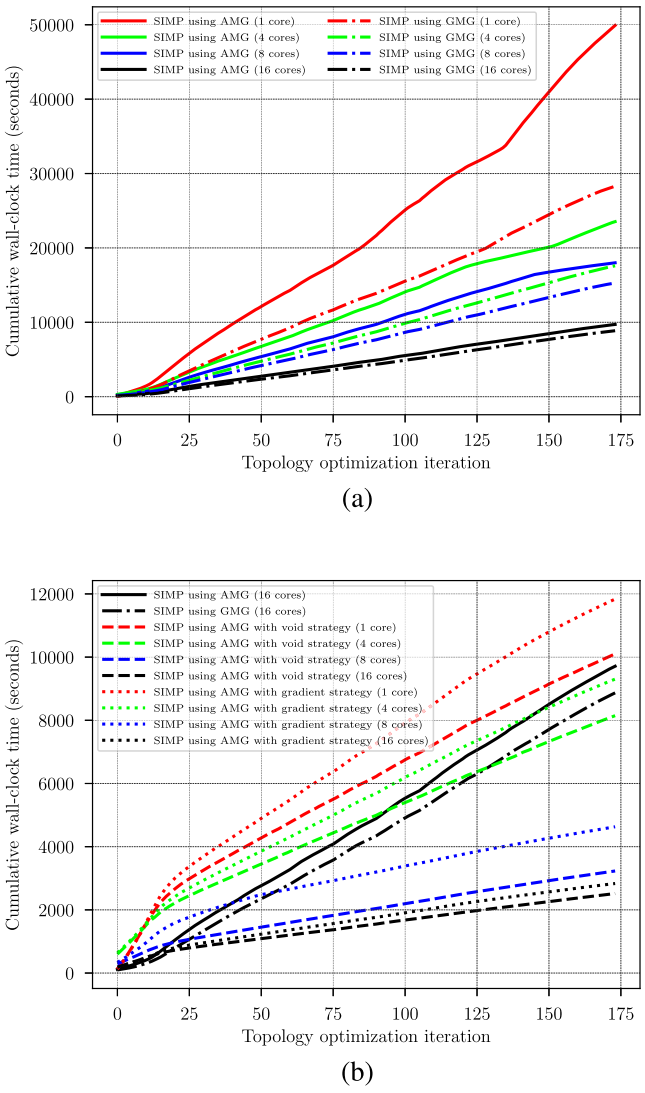

Fig. 12. Strong scaling for the cantilever experiment with 64 × 64 × 128 elements of edge size e_{s1} regularizing with $R = 2e_{s1}$: total wall-clock time (a) using AMG and GMG multigrid solvers, and (b) using AMG multigrid solver with h-refinement and coarsening using gradient- and void-based strategies.

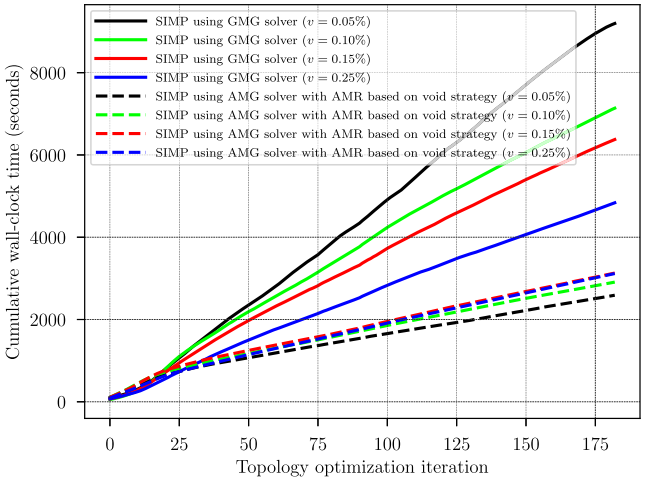

Fig. 13. Performance evaluation of cantilever experiment with 64 × 64 × 128 elements of edge size e_{s1} regularizing with $R = 2e_{s1}$ using different volume fractions for topology optimization.

strong scaling using GMG and AMG preconditioning without using AMR. We can observe that the wall-clock time decreases as increasing the computational resources preconditioning with GMG and AMG. We also note that the iterative solver preconditioned with GMG is considerably faster than the iterative solver with the AMG preconditioner using only one computing core. However, the wall-clock time is closing using both preconditioners as we increase the number of computing nodes in the workstation with an AMD Ryzen™ 9 3950X CPU. We rely on this evidence to conclude that the strong scalability on one computing node is better using the AMG preconditioner, which we combine with an AMR technique to improve the performance. Fig. 12(b) shows the cumulative wall-clock time using AMR with the resources previously mentioned. We can observe that the wall-clock time of the implementations using AMR also decreases as increasing the computational resources, obtaining an additional speedup of more than 4x for the case using 16 cores with the reference of the implementation using GMG preconditioner without AMR.

We also test the computational performance of the proposal using different volume fractions for the topology optimization of the cantilever experiment. Fig. 13 depicts the cumulative wall-clock time for the topology optimization of the cantilever model of $64 \times 64 \times 128$ elements with regularization size $R = 2e_{s1}$ using different volume fractions and maintaining the configuration detailed above. We include the same reference method used above, the topology optimization using GMG preconditioning, and for the

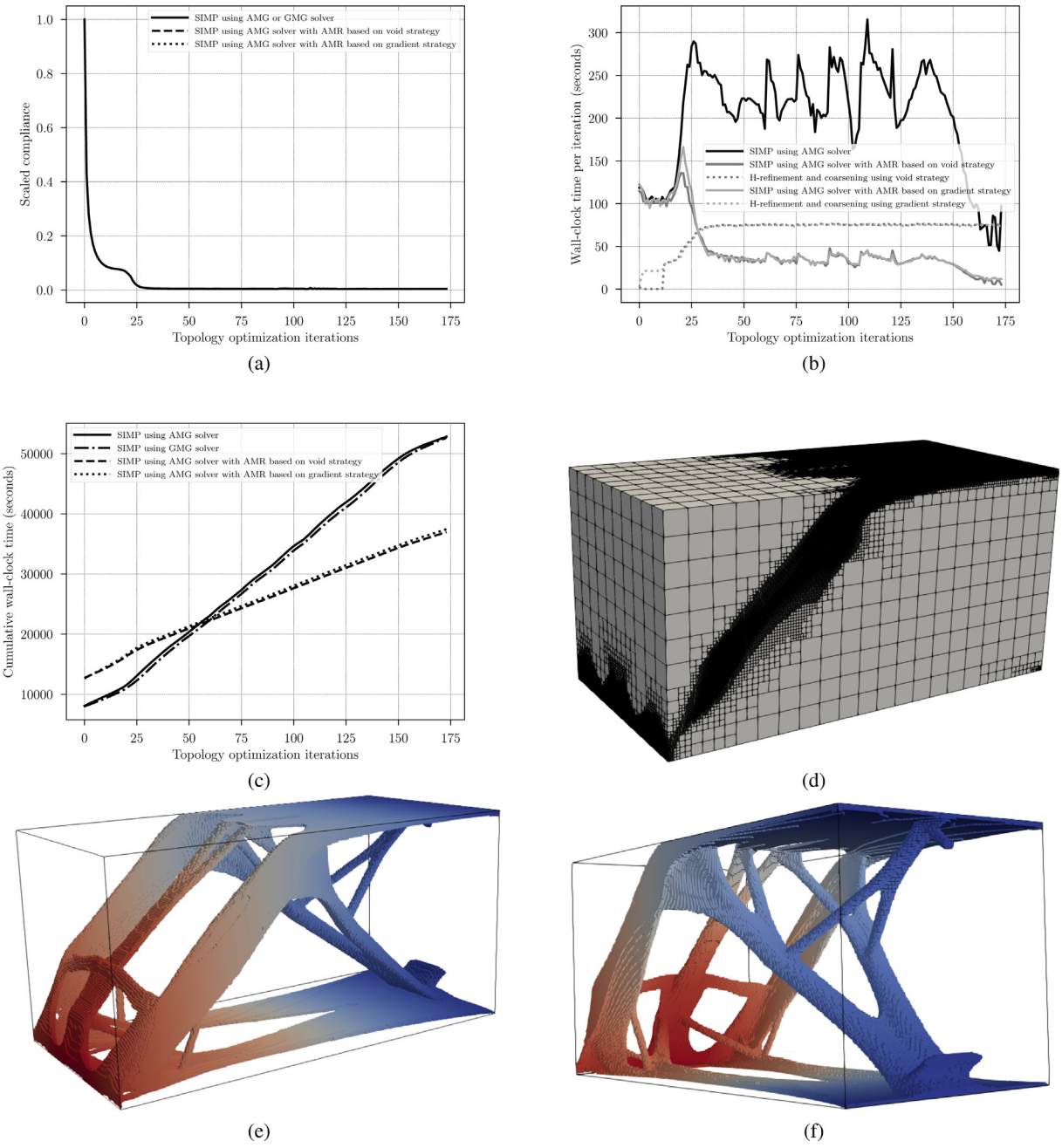

Fig. 14. The cantilever experiment with $160 \times 160 \times 320$ elements of edge size e_{s2} regularizing with $R = 2e_{s2}$ and using h-refinement and coarsening with gradient- and void-based strategies: (a) the number of finite elements used for solving, (b) wall-clock time per iteration including solving stages and dynamic refinement using 48 cores on two computing nodes, (c) total wall-clock time for the optimization using 48 cores on two computing nodes, (d) refinement detail of design variables, and (e)-(f) final design showing the displacement field.

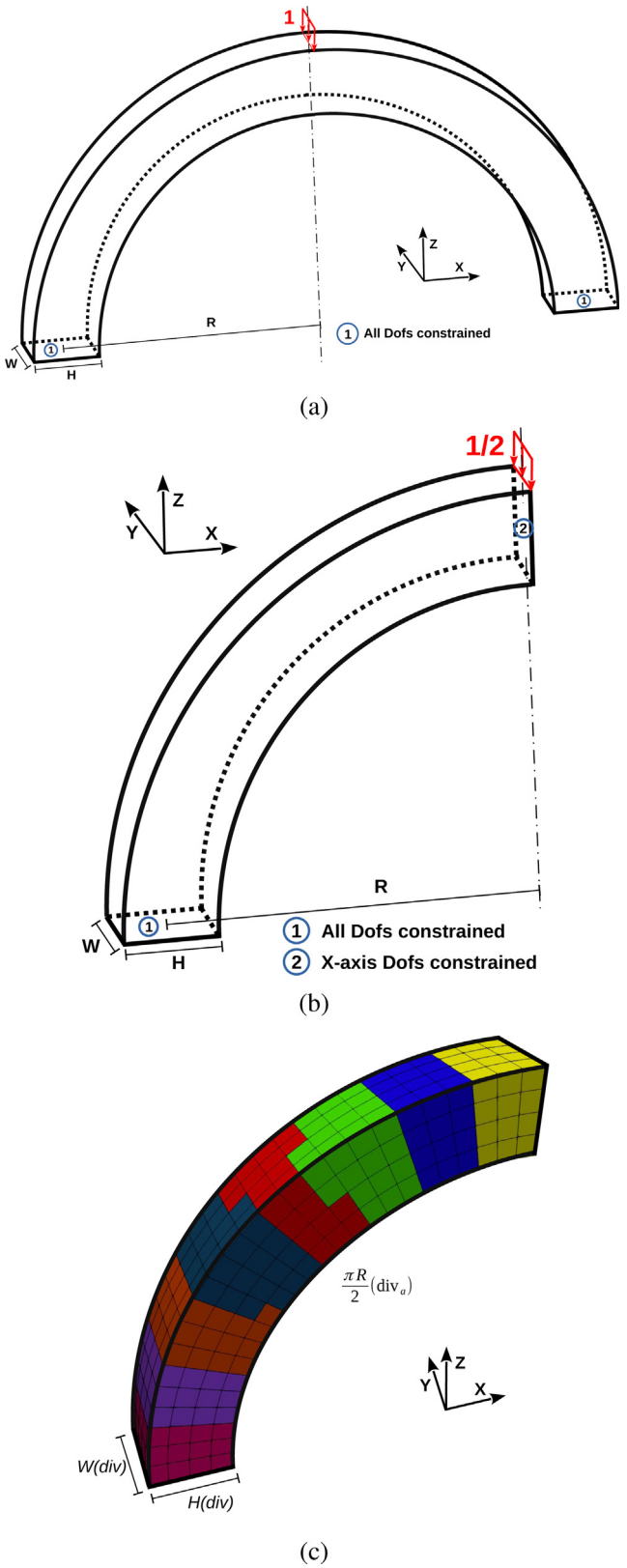

Fig. 15. The arc experiment: (a) geometric configuration, (b) symmetry simplification, and (c) mesh parameterization and partitioning into eight subdomains.

sake of clarity, the proposal using AMR with the error estimator based on the thresholding of density variables. We aim to present the limitations of the proposal. The computational advantages of using AMR reduce as the volume fraction percentage increases. We can observe that the computational improvement reduces in comparison to the reference method, which increases the computational performance by increasing the volume fraction. We attribute this behavior to the increased contrast in material properties using low volume fractions, which usually affect the GMG preconditioner due to coarsening across discontinuities, affecting the coarse grid correction.

We evaluate the scalability of the proposal using two computing nodes connected with a 10 Gigabit Ethernet network. We equip the computing nodes with two Intel Xeon E5-2687 W v4 CPUs with 12 cores allowing the running of 24 computing processes in parallel. Thus, we evaluate the proposal using 48 computing cores in the two computing nodes. Fig. 14 shows the results of the cantilever experiment with $div = 160$ elements using a radius of $R = 2e_{s2}$, with e_{s2} specified in Table 1, to regularize the density field. This tessellation generates a grid mesh of $160 \times 160 \times 320$ elements with 8192000 finite elements and 24961923 unknowns. We use the continuation strategy of the previous cantilever experiment and tune the damping factor $w_s = 0.55$ for the Jacobi smoothing to ensure convergence using GMG preconditioning. The GMG method uses four levels in the V-Cycle of Algorithm 2 from a grid with $160 \times 160 \times 320$ finite elements to a mesh of $10 \times 10 \times 20$, whereas the AMG method uses eleven levels to prolongate and restrict from a sparse coefficient matrix of 24961923×24961923 coefficient elements to a coefficient matrix of 4×4 unknowns. We use four refinement levels $l_{ref} = 4$ from the mesh of $160 \times 160 \times 320$ elements for dynamic coarsening constructing the *refinement tree*. Fig. 14(d) shows the details of the coarsening corresponding to the final design of the cantilever experiment, where we can observe the four levels of child elements with the corresponding hanging nodes. Fig. 14e) andFig. 14(f) show the details of the final design and the displacement field. We can observe that we capture many more shape details of the optimum shape compared to the optimal design shown in Fig. 10(e).

We use the same AMR thresholds $\varepsilon_v \leqslant 2E_{min}$ and $\varepsilon_g \leqslant 10^{-7}$ as the previous cantilever experiment for dynamic coarsening the mesh. Fig. 14(a) depicts the evolution of compliance during the topology optimization using different approaches with and without AMR. We can observe that the objective function evolution using GMG and AMG preconditioning and AMG preconditioning combined with dynamic AMR with different error estimators is similar. Fig. 14(b) shows the wall-clock time per iteration using 48 computing processes with and without AMR regularizing with the size $R = 2e_{s2}$. We can observe that the gradient-based approach used for AMR coarse and refine the mesh from the beginning of the experiment due to the wrong configuration of the thresholding. We also note that the computational time for the distributed coarsening and refinement operations meaningfully increases when we require network communications.

Fig. 14(c) shows the cumulative wall-clock time for all the processing of the cantilever experiment using 48 computing cores with two workstations and network communications. We observe that the elapsed time for the approaches without using AMR becomes similar, whereas it is the same for the implementations with AMR. We also note a relevant elapsed time in the first iteration of the topology optimization due to the initialization of structures needed to distribute the computation between the subdomains. This wall-clock time for the first topology optimization iteration is lower for the approaches without AMR because they only have to create the hash tables used to calculate the distributed regularization. However, the proposed method with AMR also has to compute the distributed conforming prolongation matrix $\mathbf{P_c}$ specified in Section 3. We can observe that the elapsed time to initialize the data using network communication is negligi-

ble in the cantilever experiment only using one computing node, as shown in Fig. 10(d). Thus, the wall-clock time for initializing communications can be reduced significantly using a network with higher bandwidth.

5.3. Curved beam experiment

The curved beam experiment consists of the topology optimization of a three-dimensional arc beam with fixed displacements on both sides and loads uniformly distributed in the center along the top edge. This experiment aims to show that the proposal is not subjected to any geometry constraint. Fig. 15(a) shows the geometry and boundary conditions of the finite element model. Table 1 indicates the geometric parameters and topology optimization design parameters, including the volume fraction v and the element sizes e_{s1} and e_{s2} used for regularizing the design field of the experiments using one workstation and two computing nodes. Fig. 15(b) shows the symmetry simplification used for the topology optimization, including the boundary conditions. Fig. 15(c) depicts a coarse mesh partitioned into eight subdomains using the

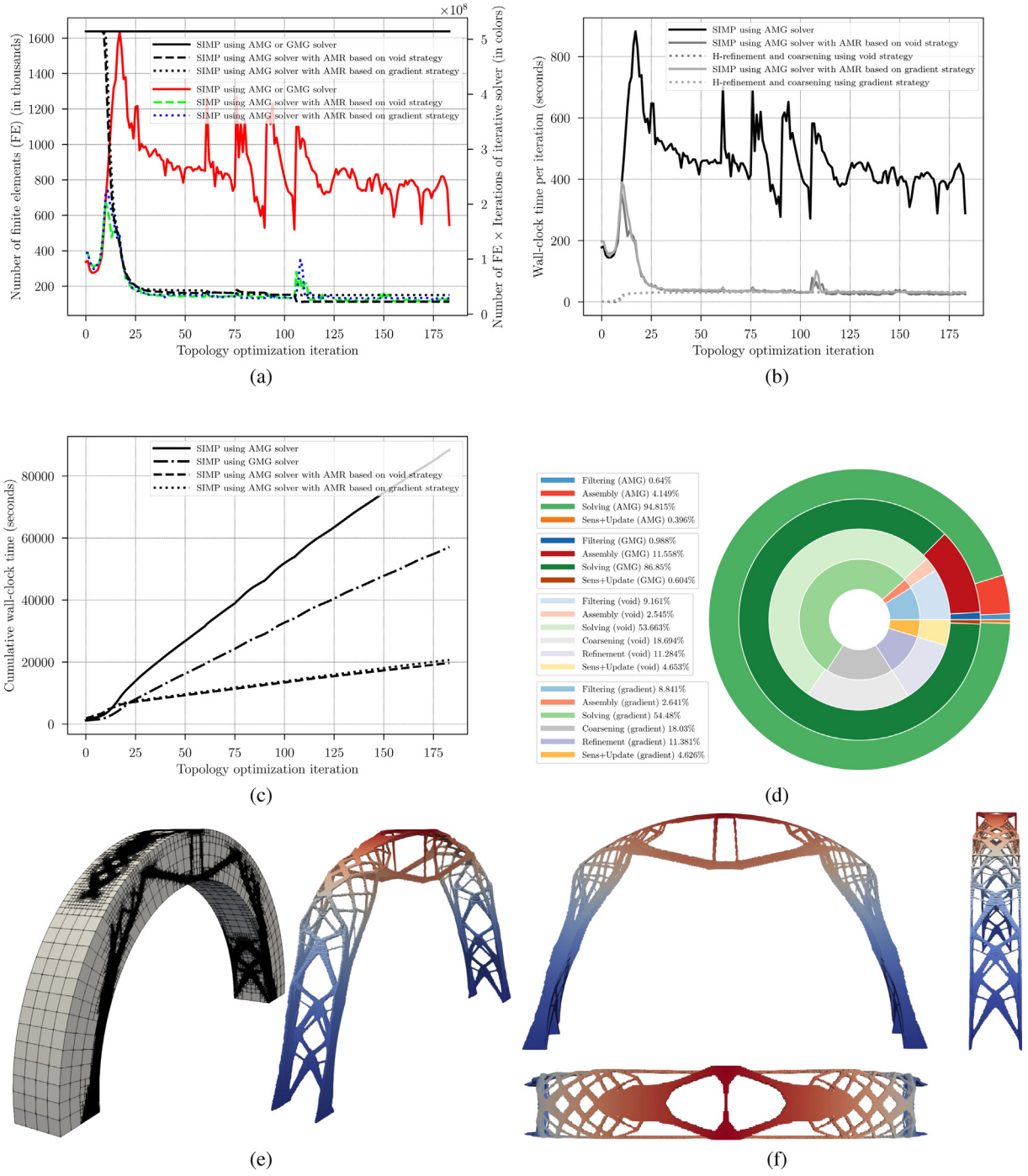

Fig. 16. The arc experiment with 64 × 64 × 400 elements of minimum edge size e_{min} regularizing with $R = 2e_{min} = 2e_{s1}$ and using h-refinement and coarsening with gradient- and void-based strategies: (a) the number of finite elements used for solving and the metric for estimating the computational cost, (b) wall-clock time per iteration including solving stages and dynamic refinement using 16 cores on one host, (c) total wall-clock time for the optimization using 16 cores on one host, (d) the percentage of wall-clock time for the different stages of the topology optimization, (e) the refinement detail of design variables, and (f) the final design showing the displacement field.

ParMetis library. We can observe that the partitioning is performed into balanced subdomains enforcing the contiguous partitions. It also shows the parameterization of the tessellation with the *div* and div_a variables. We use the number of refinement levels l_{ref} to obtain the dynamic coarsening mesh for the FEA.

Fig. 16 shows the results of the curved beam experiment with $div = 64$ and $div_a = 400$ using a radius of $R = 2e_{s1}$, with e_{s1} specified in Table 1, to regularize. This tessellation generates a model with 1638400 finite elements and 5082675 unknowns. We use one workstation with 16 computing cores for all the calculations. We initialize the projection parameters $\eta = 0.5$ and $\beta = 2.0$ of (11) following a heuristic strategy duplicating the value of β every 15 iterations from the iteration 70 until the $\beta = 64$ value. We tune the damping factor $w_s = 0.45$ for the Jacobi smoothing to ensure convergence using GMG preconditioning. The GMG method uses four levels in the V-Cycle of Algorithm 2 from a grid of $64 \times 64 \times 400$ elements to a mesh of $4 \times 4 \times 25$ elements, whereas the AMG method uses ten levels to prolongate and restrict from a sparse coefficient matrix of 5082675×5082675 coefficient elements to a coefficient matrix of 9×9 unknowns. We use four refinement levels $l_{ref} = 4$ from the grid of $64 \times 64 \times 400$ elements for dynamic coarsening constructing the *refinement tree*. We configure the thresholds $\varepsilon_v \leqslant 2E_{min}$ and $\varepsilon_g \leqslant 10^{-7}$ as criteria for dynamic coarsening the mesh using void-based and gradient-based approaches. Fig. 16(e) shows the details of the coarsening corresponding to the final design of the curved beam experiment, where we can observe the four levels of child elements with the corresponding hanging nodes. Fig. 16(f) shows the details of the final design showing the displacement field.

Fig. 16(a) shows the number of finite elements used during the topology optimization process (black color) and a metric (in colors) of the computational cost: the product of the number of finite elements and the number of iterations needed by the iterative solver to reach the prescribed tolerance. As in previous experiments, the initial iteration of topology optimization with AMR approaches uses the same amount of finite elements as the optimization without AMR, but we observe a significant reduction of the number of finite elements of almost nine times lower as topology optimization progress. We obtain a significant improvement in the metric of the computational cost by reducing the number of finite elements, which is kept during the whole topology optimization. Fig. 16(b) shows the wall-clock time per iteration for solving the curved beam experiment using 16 computing cores on one workstation with and without AMR using the size $R = 2e_{s1}$ to regularize. We can observe a significant reduction of the computing wall-clock time using AMR strategies. We also note that by using the AMR strategies, the wall-clock time for solving is more regular. We attribute this to the improvement in the conditioning of the system of equations by coarsening regions that are not of interest.

Fig. 16(c) evaluates the cumulative wall-clock time for all the processing of topology optimization with and without using AMR. We observe that the solver using GMG preconditioning is about two times faster than the resolution with the AMG preconditioner. However, we obtain a speedup of more than 4.5x and almost 3x using AMR approaches using as reference the implementation without AMR using AMG and GMG, respectively. Fig. 16(d) shows the percentage of wall-clock time for the different stages of the curved beam topology optimization with $64 \times 64 \times 400$

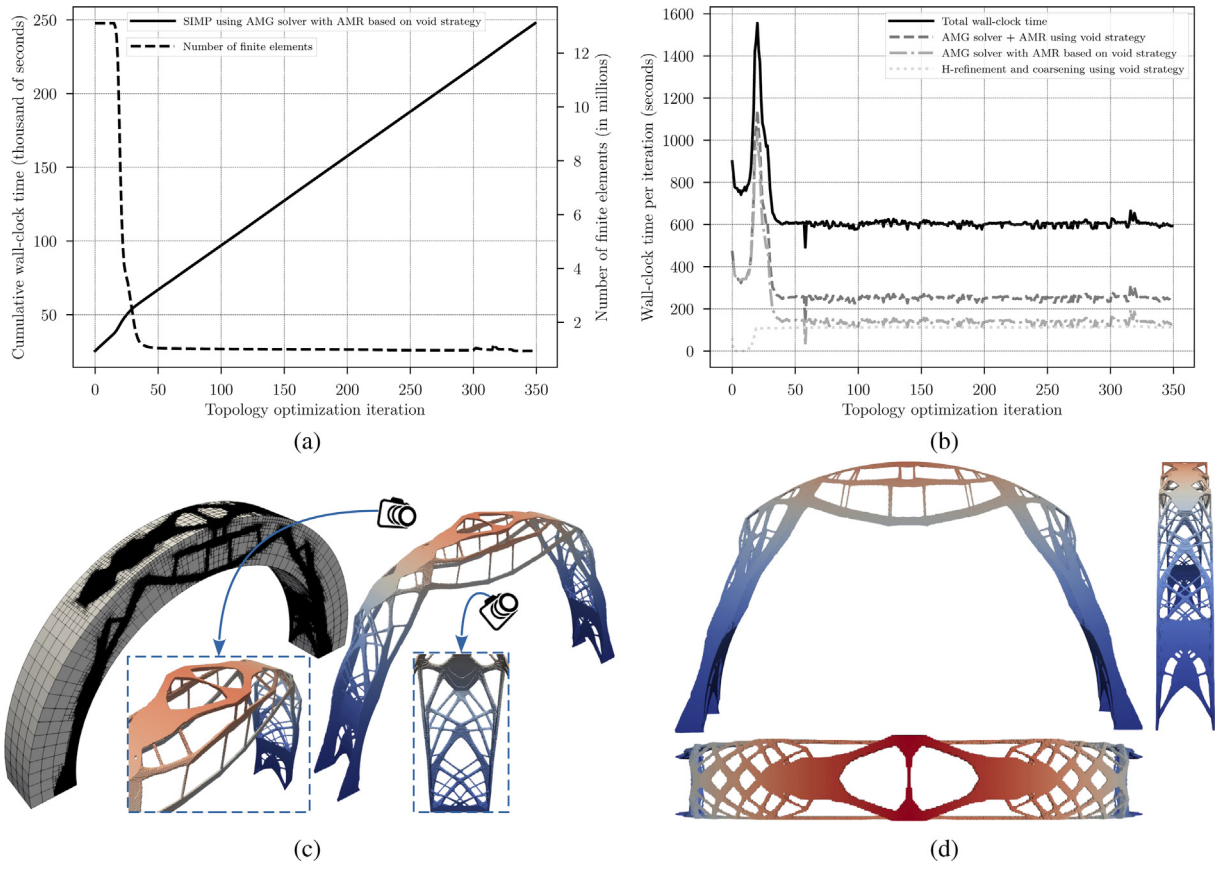

Fig. 17. Arc experiment with $128 \times 128 \times 800$ elements of minimum edge size e_{min} regularizing with $R = 2e_{min} = 2e_{s2}$ and using h-refinement and coarsening based on void strategy: (a) the number of finite elements used for solving and the cumulative wall-clock time using 48 cores on two computing nodes, (b) wall-clock time per iteration including solving stages and dynamic refinement using 48 cores on two computing nodes, (c) refinement detail of design variables for solving, and (d) the final design showing the displacement field.

elements using 16 cores on one host. We depict the information using the same format as Fig. 7. We can observe a higher reduction of the wall-clock time percentage for the assembly stage using AMR techniques than the cantilever beam experiment presented above. We also notice that the increment in the wall-clock time percentage for calculating the sensitivities and the design variable update using MMA is significantly higher than the three-dimensional experiment shown in Fig. 11. We attribute these facts to the larger size of the curved beam topology optimization.

We evaluate the scalability of the proposal with the AMR strategy using the error estimator based on void regions because we can configure it easily using Young's modulus E_{min} for weak material. We use two computing nodes with two Intel Xeon E5-2687 W v4 CPUs connected with a 10 Gigabit Ethernet network. The two computing nodes allow us to run 48 computing processes in parallel. Fig. 17 shows the results of the curved beam experiment with $div = 128$ and $div_a = 800$ using a radius of $R = 2e_{s2}$, with e_{s2} specified in Table 1, to regularize. The discretization gives rise to a model with 13107200 finite elements and 39988323 unknowns. We initialize the projection parameters $\eta = 0.5$ and $\beta = 2.0$ of (11) following a heuristic strategy duplicating the value of β every 15 iterations from iteration 200 until the $\beta = 64$ value. We use five refinement levels $l_{ref} = 5$ for dynamic coarsening constructing the *refinement tree*. We configure the threshold $\varepsilon_v \leqslant 2E_{min}$ as the criterion for dynamic coarsening the mesh using void-based approaches. Fig. 17(c) shows the details of the coarsening corresponding to the final design of the curved beam experiment, where we can observe the five levels of child elements with the corresponding hanging nodes. Fig. 17(d) shows the details of the final design showing the displacement field. We can observe that we can capture more shape details than using the tessellation of the topology shown in Fig. 16(e).

Fig. 17(a) depicts the number of finite elements used during the topology optimization process and the cumulative wall-clock time for the topology optimization using AMG preconditioning with AMR using the void-based strategy. We observe a relevant reduction of the number of finite elements. This reduction is of almost fourteen times less using five refinement levels $l_{ref} = 5$. The topology optimization takes about 70 h using the proposal. Fig. 17(b) shows the wall-clock time for solving and the total wall-clock time per iteration of topology optimization using 48 computing cores on two computing nodes using the size $R = 2e_{s2}$ to regularize. We can observe that solving is not the process consuming more computing time. The computing processes with intensive use of communication take a relevant part of the computing time. These processes are distributed regularization and adaptive coarsening and refinement. The improvement in the conditioning of the system of equations by coarsening regions that are not of interest largely compensates for the computing time consumed by communications. Nevertheless, we can reduce meaningfully the wall-clock time used by communications using a network with higher bandwidth.

5.4. Dome experiment

The dome experiment consists of the topology optimization of a three-dimensional dome of a half-sphere geometry with thin thickness. The boundary conditions consist of a vertical load applied to the top center and fixed displacements in the support. We aim to show that the proposal is not subjected to any geometry constraint evaluating its use in thin structures. We only perform the topology optimization of this experiment using the proposal with AMG preconditioning using the AMR strategy for coarsening and refinement with the error estimator based on the design variable to detect the regions of interest with void material. Fig. 18(a)

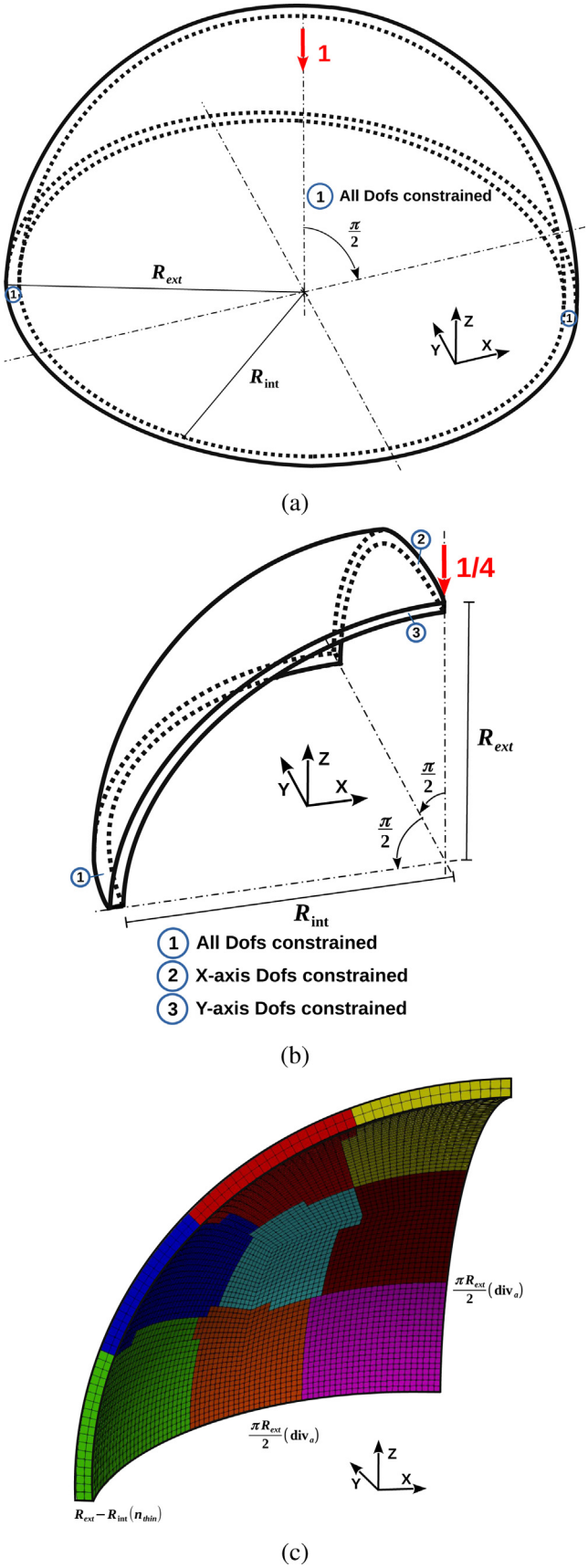

Fig. 18. The dome experiment: (a) geometric configuration, (b) symmetry simplification, and (c) mesh parameterization and partitioning into eight subdomains.

shows the geometry and boundary conditions of the finite element model. Table 1 shows the geometric and topology optimization parameters, including the volume fraction v and the element sizes e_{s1} and e_{s2} used to regularize using one workstation and two computing nodes. Fig. 18(b) shows the symmetry simplification used for the topology optimization, including the boundary conditions. We analyze one-quarter of the half-sphere geometry. Fig. 18 (c) depicts a coarse mesh partitioned into eight balanced subdomains enforcing the contiguous partitions. It also shows the parameterization of the tessellation with the n_{thin} and div_a variables. We perform the tessellation dividing the quarter of half-sphere geometry into three equal parts to generate hexahedral elements of a similar size. We can observe that we discretize elements from the central point of the one-quarter of the half-sphere geometry. We use the number of refinement levels l_{ref} to obtain the dynamic coarsening mesh for the FEA.

Fig. 19 shows the results of the dome experiment with $n_{thin} = 8$ and $div_a = 288$ using a radius of $R = 2e_{s1}$, with e_{s1} specified in Table 1, to regularize. The tessellation using three equal parts generates a mesh of $3 \times 144 \times 144 \times 8$ elements with 497664 finite elements and 1691307 unknowns. We use one workstation with 16 computing cores for all the calculations. We initialize the projection parameters $\eta = 0.3$ and $\beta = 1.0$ of (11) following a heuristic strategy duplicating the value of β every 15 iterations from iteration 200 until the $\beta = 64$ value. We use three refinement levels $l_{ref} = 3$ from the grid of $3 \times 144 \times 144 \times 8$ elements for dynamic coarsening constructing the *refinement tree*. We configure the threshold $\varepsilon_v \leqslant 2E_{min}$ as criteria for dynamic coarsening the mesh using void-based and gradient-based approaches. Fig. 19(c) shows the coarsening details corresponding to the final design of the dome experiment. We can observe the three levels of child elements with the corresponding hanging nodes. Fig. 19(d) shows the details of the final design showing the displacement field. We obtain a truss-like structure with optimal shape links between the structural elements.

Fig. 19(a) shows the number of finite elements used during the topology optimization process. We observe a reduction of five times the number of finite elements during the topology optimization progress. This reduction is lower than the previous experiments because we are only using three refinement levels $l_{ref} = 3$. It also shows the cumulative wall-clock time for all the processing of topology optimization using AMG preconditioning with AMR based on void strategy. We perform 327 iterations of topology optimization in 1 h 23' using one workstation with 16 computing cores. Fig. 19(b) shows the wall-clock time per iteration for solving using the size $R = 2e_{s1}$ to regularize. We can observe that solving dominates the computing time.

Finally, we evaluate the scalability of the proposal with the AMR strategy using the error estimator based on void regions with two computing nodes connected with a 10 Gigabit Ethernet network running 48 computing processes in parallel. Fig. 20 shows the results of the dome experiment with $n_{thin} = 16$ and $div_a = 576$ using a radius of $R = 2e_{s2}$, with e_{s2} specified in Table 1, to regularize. The tessellation generates a model with 3981312 finite elements and 12734547 unknowns. We initialize the projection parameters $\eta = 0.3$ and $\beta = 1.0$ of (11) following a heuristic strategy duplicating the value of β every 15 iterations from iteration 200 until the

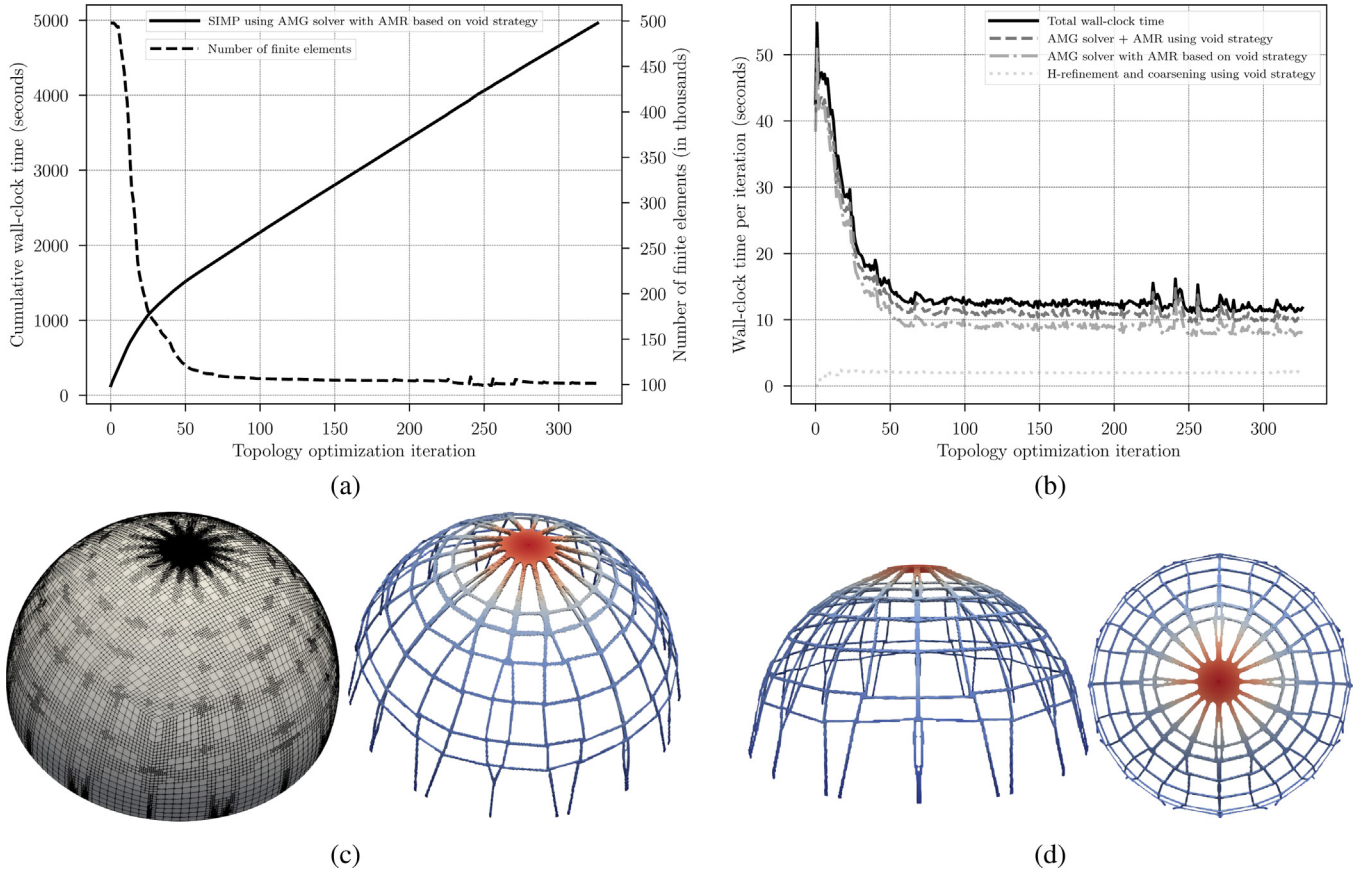

Fig. 19. The dome experiment with $3 \times 144 \times 144 \times 8$ elements of minimum edge size e_{min} regularizing with $R = 2e_{min}=2e_{s1}$ and using h-refinement and coarsening based on void strategy: (a) the number of finite elements used for solving and the cumulative wall-clock time using 16 cores on one host, (b) wall-clock time per iteration including solving stages and dynamic refinement using 16 cores on one host, (c) refinement detail of design variables for solving, and (d) the final design showing the displacement field.

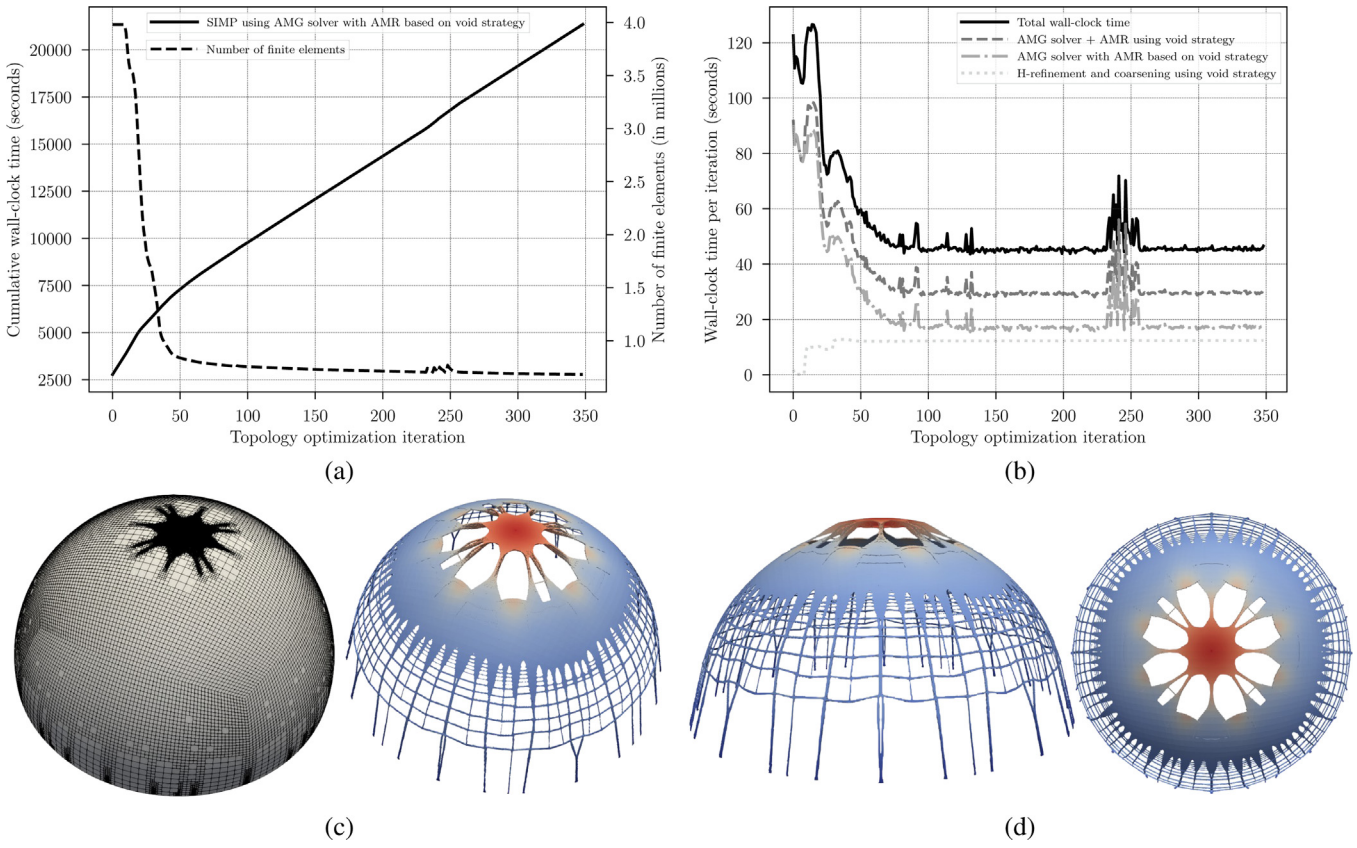

Fig. 20. The dome experiment with $3 \times 288 \times 288 \times 16$ elements of minimum edge size e_{min} regularizing with $R = 2e_{min}$=$2e_{s2}$ and using h-refinement and coarsening based on void strategy: (a) the number of finite elements used for solving and the cumulative wall-clock time using 48 cores on two computing nodes, (b) wall-clock time per iteration including solving stages and dynamic refinement using 48 cores on two computing nodes, (c) refinement detail of design variables for solving, and (d) final design showing the displacement field.

$\beta = 64$ value. We use four refinement levels $l_{ref} = 4$ for dynamic coarsening constructing the *refinement tree*. We configure the threshold $\varepsilon_v \leqslant 2E_{min}$ as the criterion for dynamic coarsening the mesh using void-based approaches.

Fig. 20(c) shows the details of the coarsening in the optimized design, where we can observe the four levels of child elements with the corresponding hanging nodes. Fig. 20(d) shows the details of the optimized design showing the displacement field. We can observe that the optimal design obtained is more similar to a shell-like structure with variable thickness [74], whereas the design obtained with a coarse finite element mesh has an appearance of many truss-like designs, as shown in Fig. 19(d). We attribute this effect to the artificial length-scale limitation caused by the use of a coarse mesh.

Fig. 20(a) shows the number of finite elements used during the topology optimization process. The reduction of finite elements during the topology optimization progress is about seven times the initial mesh by dynamic coarsening using four refinement levels $l_{ref} = 4$. We also show the cumulative wall-clock time for all the processing of topology optimization using AMG preconditioning with AMR based on void strategy. We perform 350 iterations of topology optimization in 6 h using 48 computing cores using two workstations connected with a 10 Gigabit Ethernet network. Fig. 20(b) shows the wall-clock time per iteration for solving using the size $R = 2e_{s2}$ to regularize. We can observe a relevant difference in the ratio of computing time for solving to total computing time per topology optimization iteration. We attribute this difference to the computing time of processes requiring intensive use of communications. These processes are distributed regularization and adaptive coarsening and refinement. As previously mentioned, we can increase the data-sharing performance by using a network with higher bandwidth.

6. Conclusion and future works

We present an efficient implementation of density-based topology optimization combining dynamic AMR and parallel computing for distributed memory systems. We focus on the acceleration of the bottleneck of the topology optimization process, the analysis using FEA, using dynamic coarsening to provide an equivalent design to obviating the AMR method. The proposal consists of optimizing with a fine mesh and analyzing using a dynamic coarser mesh. We calculate the sensitivities and update the variables in the fine mesh, and thus the coarsened areas do not affect the evolution of the optimization. We use an error estimator based on the design variables, identifying the regions of interest by thresholding using the parameters of the power-low interpolation function for material penalization of the SIMP method. The underline idea is that the computation of areas with void material contributes significantly to the overall computing time but little to the accuracy of the optimized design.

We evaluate the performance and scalability of the proposal using different computational resources. In particular, a different number of computing cores and distributed computing hosts. We check the use of multigrid methods for preconditioning the Krylov subspace iteration solver used in this work: a distributed conjugate gradient method. We perform these experiments for two- and

three-dimensional topology optimization problems. We also test using different error estimators for the AMR approach being the criterion based on design variables the most robust for reducing the computational cost. We have to remark that we perform all the experiments using a reduced distance to regularize. For higher regularization distances, it is more suitable to use a PDE for filtering the design field. We also have to remark that we obtain the computing time presented in this work using computing resources that are not of the latest generation, and thus faster results could be obtained using other computational resources.

We obtain a significant increment of computing performance for the overall computing time using AMR techniques. However, the computing processes with intensive use of communication can take a relevant part of the computing time when we use distributed computing with many computing hosts. In these cases, we require a network with higher bandwidth than the network used in the experiments. We also note that the distributed coarsening and refinement method can have a relevant computational cost using a large number of refinement levels, and thus we have to find a trade-off between the cost of solving and the computational requirements to reduce the system of equations to solve. Nevertheless, the improvement in the conditioning of the system of equations by coarsening void material usually makes up for the computing time consumed by AMR and intensive use of communications.

We have to mention the introduction of solution-based error estimators for stress-constrained topology optimization using AMR as future work. The use of the MMA method allows us to manage stress constraints in the topology optimization formulation, which is relevant for structural applications. The GMG preconditioner shows better properties than the AMG preconditioner for the three-dimensional problems presented in this work. For this reason, we also have to mention the development of a GMG preconditioner for the Krylov subspace iteration solver using AMR as future work using the presented error estimation strategies. Finally, we propose the extension of the distributed coarsening and refinement approach to the use of simplicial meshes as future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been supported by the AEI/FEDER and UE under the contract DPI2016-77538-R.

References

[1] Deaton J, Grandhi R. A survey of structural and multidisciplinary continuum topology optimization: post 2000. Multidiscip Optim 2000;49(2014):1–38.
[2] Ribeiro T, Bernardo L, Andrade J. Topology Optimisation in Structural Steel Design for Additive Manufacturing. Appl Sci 2021;11:2112.
[3] Bendsøe MP, Kikuchi N. Generating optimal topologies in structural design using a homogenization method. Comput Meth Appl Mech Eng 1988;71:197–224.
[4] Zhou M, Rozvany GIN. The COC algorithm, Part II: Topological, geometrical and generalized shape optimization. Comput Methods Appl Mech Eng 1991;89:309–36.
[5] Bendsøe MP, Sigmund O. Topology Optimization – Theory, Methods, and Applications. second ed. Berlin Heidelberg: Springer-Verlag; 2004.
[6] Tsavdaridis KD, Kingman JJ, Toropov VV. Application of structural topology optimisation to perforated steel beams. Comput Struct 2015;158:108–23.
[7] Saadlaoui Y, Milan JL, Rossi JM, Chabrand P. Topology optimization and additive manufacturing: Comparison of conception methods using industrial codes. J Manuf Syst 2017;43:178–86.
[8] Plocher J, Panesar A. Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures. Mater Des 2019;183:108164.
[9] Van Truong T, Kureemun U, Tan VBC, Lee HP. Study on the structural optimization of a flapping wing micro air vehicle. Struct Multidisc Optim 2018;57:653–64.
[10] Bagherinejad MH, Haghollahi A. Study on Topology Optimization of Perforated Steel Plate Shear Walls in Moment Frame Based on Strain Energy. Int J Steel Struct 2020;20:1420–38.
[11] Choi WH, Kim JM, Park GJ. Comparison study of some commercial structural optimization software systems. Struct Multidisc Optim 2016;54:685–99.
[12] Liu Z, Cease H, Collins JT, Nudell J, Preissner CA. Optimization for the APS-U Magnet Support Structur., In: Proc. MEDSI'16, Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference, Barcelona, Spain; 2017. p. 254–6.
[13] Zhang S, Li P, Zhong Y, Xiang J. Structural Topology Optimization Based on the Level Set Method Using COMSOL. Comput Model Eng Sci 2014;101:17–31.
[14] Allaire G, Jouve F, Toader AM. Structural optimization using shape sensitivity analysis and a level-set method. J Comput Phys 2004;194:363–93.
[15] Dambrine M, Kateb D. On the ersatz material approximation in level-set methods, ESAIM: Control. Optim Calculus Variat 2010;16:618–34.
[16] Venkataraman S, Haftka RT. Structural optimization complexity: what has Moore's law done for us? Struct Multidiscip Optim 2004;28:375–87.
[17] Wang S, De Sturler E, Paulino GH. Large-scale topology optimization using preconditioned Krylov subspace methods with recycling. Int J Numer Methods Eng 2007;69:2441–68.
[18] Amir O, Stolpe M, Sigmund O. Efficient use of iterative solvers in nested topology optimization. Struct Multidiscip Optim 2010;42:55–72.
[19] Amir O, Aage N, Lazarov BS. On multigrid-CG for efficient topology optimization. Struct Multidiscip Optim 2014;49:815–29.
[20] Amir O, Bendsøe MP, Sigmund O. Approximate reanalysis in topology optimization. Int J Numer Methods Eng 2009;78:1474–91.
[21] Nguyen TH, Paulino GH, Song J, Le CH. A computational paradigm for multiresolution topology optimization (MTOP). Struct Multidiscip Optim 2010;41:525–39.
[22] Liu H, Wang Y, Zong H, Wang MY. Efficient structure topology optimization by using the multiscale finite element method. Struct Multidiscip Optim 2018;58:1411–30.
[23] Gupta DK, van Keulen F, Langelaar M. Design and analysis adaptivity in multiresolution topology optimization. Int J Numer Methods Eng 2020;121:450–76.
[24] Borrvall T, Petersson J. Large-scale topology optimization in 3D using parallel computing. Comput Methods Appl Mech Eng 2001;190:6201–29.
[25] Vemaganti K, Lawrence WE. Parallel methods for optimality criteria-based topology optimization. Comput Methods Appl Mech Eng 2005;194:3637–67.
[26] Aage N, Andreassen E, Lazarov BS. Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. Struct Multidiscip Optim 2015;51:565–72.
[27] Aage N, Andreassen E, Lazarov BS, Sigmund O. Giga-voxel computational morphogenesis for structural design. Nature 2017;550:84–6.
[28] Liu H, Tian Y, Zong H, Ma Q, Wang MY, Zhang L. Fully parallel level set method for large-scale structural topology optimization. Comput Struct 2019;221:13–27.
[29] Martínez-Frutos J, Martínez-Castejón PJ, Herrero-Peréz D. Fine-grained GPU implementation of assembly-free iterative solver for finite element problems. Comput Struct 2015;157:9–18.
[30] Martínez-Frutos J, Herrero-Peréz D. Large-scale robust topology optimization using multi-GPU systems. Comput Methods Appl Mech Eng 2016;311:393–414.
[31] Martínez-Frutos J, Herrero-Pérez D. GPU acceleration for evolutionary topology optimization of continuum structures using isosurfaces. Comput Struct 2017;182:119–36.
[32] Martínez-Frutos J, Martínez-Castejón PJ, Herrero-Pérez D. Efficient topology optimization using GPU computing with multilevel granularity. Adv Eng Softw 2017;106:47–62.
[33] Herrero-Pérez D, Martínez-Castejón PJ. Multi-GPU acceleration of large-scale density-based topology optimization. Adv Eng Softw 2021;157–158:103006.
[34] Olm M, Badia S, Martín A. On a general implementation of h- and p-adaptive curl-conforming finite elements. Adv Eng Soft 2019;132:74–91.
[35] Wang Y, Kang Z, He Q. Adaptive topology optimization with independent error control for separated displacement and density fields. Comput Struct 2014;135:50–61.
[36] de Troya MS, Tortorelli D. Adaptive mesh refinement in stress-constrained topology optimization. Struct Multidiscip Optim 2018;58:2369–86.
[37] Salazar de Troya M, Tortorelli D. Three-dimensional adaptive mesh refinement in stress-constrained topology optimization. Struct Multidiscip Optim 2020;62:2467–79.
[38] Baiges J, Matínez-Frutos J, Herrero-Pérez D, Otero F, Ferrer A. Large-scale stochastic topology optimization using adaptive mesh refinement and coarsening through a two-level parallelization scheme. Comput Methods Appl Mech Eng 2019;343:186–206.
[39] De Sturler E, Paulino GH, Wang S. Topology optimization with adaptive mesh refinement. In: Int. Conf. on Computation of Shell and Spatial Structures (IASS-IACM 2008), Ithaca, NY, USA; 2008. p. 1–4.

[40] Kikuchi N, Chung KY, Torigaki T, Taylor JE. Adaptive finite element methods for shape optimization of linearly elastic structures. Comp Meth Appl Mech Eng 1986;57:67–89.
[41] Costa JCA, Alves MC. Layout optimization with h-adaptivity of structures. Int J Numer Meth Eng 2003;58:83–102.
[42] Stainko R. An adaptive multilevel approach to the minimal compliance problem in topology optimization. Commun Numer Meth Eng 2006;22:109–18.
[43] Borrvall T, Petersson J. Topology optimization using regularized intermediate density control. Comput Methods Appl Mech Eng 2001;190:4911–28.
[44] Bruggi M, Verani M. A fully adaptive topology optimization algorithm with goal-oriented error control. Comput Struct 2011;89:1481–93.
[45] Liu H, Hu Y, Zhu B, Matusik W, Sifakis E. Topology optimization using regularized intermediate density control. ACM Trans Graph 2018;37:1–14.
[46] Marco O, Ródenas J, Navarro-Jiménez J, Tur M. Robust h-adaptive meshing strategy considering exact arbitrary CAD geometries in a Cartesian grid framework. Comput Struct 2017;193:87–109.
[47] Muñoz D, Albelda J, Ródenas J, Nadal E. Improvement in 3d topology optimization with h-adaptive refinement using the cartesian grid finite element method. Int J Numer Meth Eng; 2021. 1–28.
[48] Li H, Yamada T, Jolivet P, Furuta K, Kondoh T, Izui K, Nishiwaki S. Full-scale 3d structural topology optimization using adaptive mesh refinement based on the level-set method. Finite Elem Anal Des 2021;194:103561.
[49] Červený J, Dobrev V, Kolev T. Nonconforming mesh refinement for high-order finite elements. SIAM J Sci Comput 2019;41:C367–92.
[50] Bendsøe MP, Sigmund O. Material interpolation schemes in topology optimization. Arch Appl Mech 1999;69:635–54.
[51] Sigmund O, Petersson J. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. Struct Optim 1998;16:68–75.
[52] Xu S, Cai Y, Cheng G. Volume preserving nonlinear density filter based on heaviside functions. Struct Multidiscip Optim 2010;41:495–505.
[53] Bourdin B. Filters in topology optimization. Int J Numer Methods Eng 2001;50:2143–58.
[54] Guest J, Prevost J, Belytschko T. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. Int J Numer Meth Eng 2004;61:238–54.
[55] Sigmund O. Morphology-based black and white filters for topology optimization. Struct Multidiscip Optim 2007;33:401–24.
[56] Wang F, Lazarov B, Sigmund O. On projection methods, convergence and robust formulations in topology optimization. Struct Multidiscip Optim 2011;43:767–84.
[57] Svanberg K. The method of moving asymptotes-a new method for structural optimization. Int J Numer Methods Eng 1987;24:359–73.
[58] Aage N, Lazarov B. Parallel framework for topology optimization using the method of moving asymptotes. Struct Multidisc Optim 2013;47:493–505.
[59] MFEM, Modular finite element methods, http://mfem.org; 2021.
[60] Hypre, A library of high performance preconditioners, http://www.llnl.gov/CASC/hypre/; 2021.
[61] Lambe A, Czekanski A. Topology optimization using a continuous density field and adaptive mesh refinement. Int J Numer Meth Eng 2018;113:357–73.
[62] Nana A, Cuillière J, Francois V. Towards adaptive topology optimization. Adv Eng Soft 2016;100:290–307.
[63] Bitzarakis S, Papadrakakis M, Kotsopulos A. Parallel solution techniques in computational structural mechanics. Comput Methods Appl Mech Eng 1997;148:75–104.
[64] Karypis G, Kumar V. Multilevel k-way partitioning scheme for irregular graphs. J Parallel Dist Com 1998;48:96–129.
[65] Karypis G, Schloegel K. ParMeTis: Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 4.0, Technical Report, University of Minnesota, Minneapolis, MN; 2013.
[66] Stüben K. A review of algebraic multigrid. J Comput Appl Math 2001;128:281–309.
[67] Hülsemann F, Kowarschik M, Mohr M, Rüde U. Parallel geometric multigrid. In: Bruaset AM, Tveito A, editors. Numerical Solution of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering, vol. 51. Heidelberg: Berlin; 2006. p. 165–208.
[68] Henson V, Yang U. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. Appl Numer Math 2002;41:155–77.
[69] Yang U. Parallel Algebraic Multigrid Methods – High Performance Preconditioners. In: Bruaset AM, Tveito A, editors. Numerical Solution of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering, vol. 51. Heidelberg: Berlin; 2006. p. 209–36.
[70] Ruge J, Stüben K. Algebraic Multigrid. In: Ewing R, editor. Multigrid Methods, Society for Industrial and Applied Mathematics (SIAM), 3600 University City Science Center. Philadelphia: Pennsylvania; 1987. p. 73–130.
[71] De Sterck H, Falgout R, Nolting J, Yang U. Distance-two interpolation for parallel algebraic multigrid. Numer Linear Algebra Appl 2008;15:115–39.
[72] Chow E. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. SIAM J Sci Comput 2000;21:1804–22.
[73] Lazarov BS, Sigmund O. Filters in topology optimization based on Helmholtz-type differential equations. Int J Numer Methods Eng 2011;86:765–81.
[74] Sigmund O, Aage N, Andreassen E. On the (non-)optimality of Michell structures. Struct.Multidiscip Optim 2016;54:361–73.