Comput Mech (2011) 47:265–282
DOI 10.1007/s00466-010-0546-5

ORIGINAL PAPER

Parallel simulations of three-dimensional cracks using the generalized finite element method

D.-J. Kim · C. A. Duarte · N. A. Sobh

Received: 10 June 2010 / Accepted: 8 October 2010 / Published online: 27 October 2010

Abstract This paper presents a parallel generalized finite element method (*GFEM*) that uses customized enrichment functions for applications where limited a priori knowledge about the solution is available. The procedure involves the parallel solution of local boundary value problems using boundary conditions from a coarse global problem. The local solutions are in turn used to enrich the global solution space using the partition of unity methodology. The parallel computation of local solutions can be implemented using a single pair of scatter–gather communications. Several numerical experiments demonstrate the high parallel efficiency of these computations. For problems requiring non-uniform mesh refinement and enrichment, load unbalance is addressed by defining a larger number of small local problems than the number of parallel processors and by sorting and solving the local problems based on estimates of their workload. A simple and effective estimate of the largest number of processors where load balance among processors is maintained is also proposed. Several three-dimensional fracture mechanics problems aiming at investigating the accuracy and parallel performance of the proposed *GFEM* are analyzed.

Submitted to Computational Mechanics.

D.-J. Kim
Department of Architectural Engineering, Kyung Hee University, Engineering Building, 1 Sochon-Dong Kihung-Gu, Kyunggi-Do, Yongin 446-701, Korea

C. A. Duarte (✉) · N. A. Sobh
Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Newmark Laboratory, 205 North Mathews Avenue, Urbana, IL 61801, USA
e-mail: caduarte@uiuc.edu

Keywords Generalized FEM · Extended FEM · Parallel computation · OpenMP · Global–local analysis · Fracture

1 Introduction

Realistic simulations of many practical fracture mechanics problems are still formidable tasks for the finite element method (*FEM*) [38]. The accurate solution of three-dimensional fracture mechanics problems requires aggressive mesh refinement and polynomial enrichment around crack fronts. This creates difficulties in the parallel solution of the problem since load balancing becomes non-trivial. Typical parallel *FEM* implementations partition the computational domain and distribute the partitions among processors such that each processor processes the same computational load [15–19,22]. However, *FEM* discretizations with non-uniform element sizes and/or polynomial orders are difficult to partition since estimating the computational load of each partition is not trivial [26]. Furthermore, the computation load in each partition changes during the simulation as the *FEM* mesh is adapted and/or the crack propagates. The use of massive computational power by itself is not sufficient. Instead, advances in existing techniques and the development of scalable algorithms for this class of problems are needed.

The generalized or extended finite element method (*G/XFEM*) [2–4,8,25,27,35,36] allows straightforward construction of discrete solution spaces using non-polynomial functions while preserving the sparsity of global matrices. These functions are chosen carefully to mimic known properties of the function to be approximated like discontinuities or singularities. Most applications of these methods have relied on analytically derived enrichment functions. However, while these functions add flexibility and robustness to these methods, they are in general not able to deliver accurate solutions on coarse three-dimensional meshes. To overcome this limitation, local mesh refinement must be performed as in the standard finite element method [14,31]. This creates several of the drawbacks of the *FEM* with remeshing, in particular, low scalability in a parallel environment.

In [9,7,10,20,21] we presented a generalized finite element method based on the solution of interdependent global (structural) and local (crack) scale problems. The local problems focus on the resolution of fine scale features of the solution in the vicinity of three-dimensional cracks while the global problem addresses the macro-scale structural behavior. The local solutions are embedded into the global solution space using the partition of unity method. The local problems are accurately solved using an *hp-GFEM* for three-dimensional cracks [31,32] and thus the method does not rely on analytical solutions. The methodology enables accurate modeling of three-dimensional cracks on meshes with elements that are orders of magnitude larger than those required by the *FEM* or previously available *GFEM*. As a result, a single global mesh can be used to analyze any crack configuration [21] or multiple interacting cracks [20]. Furthermore, only a few degrees of freedom are hierarchically added to the global (coarse-scale) discretization, regardless of the number of degrees of freedom required to solve the local problems [9]. We denote this class of methods as a *GFEM* with global–local enrichment functions ($GFEM^{gl}$). Global–local enrichment functions also enable the analysis of problems with sharp thermal gradients using coarse meshes, as demonstrated in [28,29].

In this paper, we formulate the $GFEM^{gl}$ such that the bulk of the computations can be efficiently done in parallel. In this approach, a local problem is defined for each node of the global mesh whose shape function support intersects a region of interest—like a neighborhood of a crack. These local problems can be efficiently solved in parallel, since no communication among processors solving different local problems is required. In the parallel $GFEM^{gl}$ presented here, load balancing among processors is addressed by defining a larger number of local problems than the number of parallel processors, and by sorting and solving the local problems based on estimates of their workload. A simple and effective estimate of the largest number of processors below which load balance among processors is maintained is also presented.

The remainder of this paper is organized as follows. Sections 2 and 3 review the *GFEM* and in particular the *GFEM* with global–local enrichment functions presented in [21]. In Sect. 4, an approach to handle the numerical integration on global elements enriched with global–local enrichment functions computed at different local problems is presented. The implementation of the proposed parallel $GFEM^{gl}$ using the OpenMP (Open Multi-Processing) programming model [30] and a sorting algorithm to improve the computational load balancing among processors are also discussed in Sect. 4. In Sect. 5, numerical examples demonstrating the accuracy and the parallel efficiency of the methodology are presented. Finally, the conclusions of this paper are drawn in Sect. 6.

2 Generalized finite element method: a summary

The generalized FEM [2,3,8,27,35] is an instance of the so-called partition of unity method (PUM) which has its origins in the works of Babuška et al. [1,3,24] and Duarte and Oden [6,11–13,27]. The generalized FEM (*GFEM*) denotes a PUM with the partition of unity provided by Lagrangian finite element shape functions. The same method is also known as the extended FEM (*XFEM*) [4,25]. A recent review of Generalized/eXtended FEMs along with a brief history of their developments can be found in [5].

Generalized FEM approximation spaces (i.e., trial spaces) consist of three components—(a) patches or clouds, (b) a partition of unity, and (c) the patch or cloud approximation spaces. We describe these components as follows:

(a) *Patches or clouds* ω_α In the generalized finite element method, a cloud ω_α is given by the union of the finite elements sharing node α of the finite element mesh covering the domain of interest Ω. The set $\{\omega_\alpha\}_{\alpha=1}^N$, in a finite element mesh with N nodes, is an open cover of Ω, i.e., $\Omega = \cup_{\alpha=1}^N \omega_\alpha$.

(b) *Partition of unity subordinate to the cover* $\{\omega_\alpha\}_{\alpha=1}^N$ The Lagrangian finite element shape functions φ_α, $\alpha = 1, \ldots, N$, constitute a partition of unity, i.e., $\sum_{\alpha=1}^N \varphi_\alpha(\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$ in Ω. This is a key property used in partition of unity methods.

(c) *Cloud approximation spaces* χ_α To each cloud ω_α, we associate a $D_L(\alpha)$-dimensional space χ_α of functions defined on ω_α, namely,

$$\chi_\alpha = \operatorname{span}\{L_{\alpha i},\ 1 \le i \le D_L(\alpha),\ L_{\alpha i} \in H^1(\omega_\alpha)\}.$$

The basis functions $L_{\alpha i}$ above are also known as *enrichment functions*. A cloud approximation $\boldsymbol{u}_\alpha^{hp}(\boldsymbol{x}) \in \chi_\alpha$ of $\boldsymbol{u}|_{\omega_\alpha}$—the restriction to ω_α of a function $\boldsymbol{u}$ defined on Ω—can be written as

$$\boldsymbol{u}_\alpha^{hp}(\boldsymbol{x}) = \sum_{i=1}^{D_L} \underline{\boldsymbol{u}}_{\alpha i} L_{\alpha i}(\boldsymbol{x})$$

where $\underline{\boldsymbol{u}}_{\alpha i}$, $i = 1, \ldots, D_L(\alpha)$, are degrees of freedom.

The trial space for the *GFEM* is given by

$$\boldsymbol{X}(\Omega) \equiv \sum_{\alpha=1}^N \varphi_\alpha \chi_\alpha = \operatorname{span}\{\phi_{\alpha i} := \varphi_\alpha L_{\alpha i},\ 1 \le i \le D_L(\alpha),\ 1 \le \alpha \le N\} \qquad (1)$$

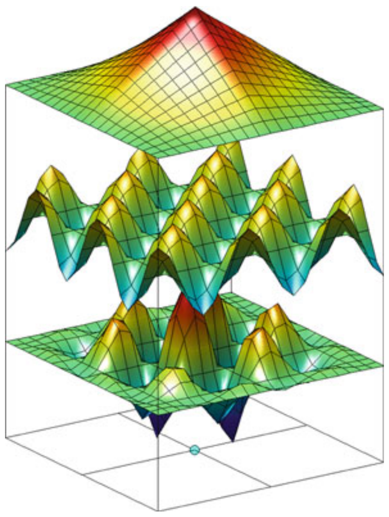


Fig. 1 Construction of a generalized FEM shape function. Here, φ_α is the function at the *top*, $L_{\alpha i}$ is the function in the *middle* and the generalized FE shape function, $\phi_{\alpha i}$, is shown at the *bottom*

The function

$$\phi_{\alpha i}(\boldsymbol{x}) = \varphi_\alpha(\boldsymbol{x}) L_{\alpha i}(\boldsymbol{x}) \quad \text{(no summation on } \alpha), \tag{2}$$

where α is a node in the finite element mesh, is called a *GFEM* shape function. Figure 1 illustrates the construction of *GFEM* shape functions in a two-dimensional domain.

A *GFEM* approximation $\boldsymbol{u}^{hp}(\boldsymbol{x}) \in X(\Omega)$ of a vector value function $\boldsymbol{u}$ can be written as

$$\begin{aligned}\boldsymbol{u}^{hp}(\boldsymbol{x}) &= \sum_{\alpha=1}^{N}\sum_{i=1}^{D_L} \underline{\boldsymbol{u}}_{\alpha i}\phi_{\alpha i}(\boldsymbol{x}) = \sum_{\alpha=1}^{N}\sum_{i=1}^{D_L} \underline{\boldsymbol{u}}_{\alpha i}\varphi_\alpha(\boldsymbol{x})L_{\alpha i}(\boldsymbol{x}) \\ &= \sum_{\alpha=1}^{N}\varphi_\alpha(\boldsymbol{x})\sum_{i=1}^{D_L}\underline{\boldsymbol{u}}_{\alpha i}L_{\alpha i}(\boldsymbol{x}) = \sum_{\alpha=1}^{N}\varphi_\alpha(\boldsymbol{x})\boldsymbol{u}_\alpha^{hp}(\boldsymbol{x})\end{aligned}$$

3 Parallel solution of two-scale problems using global–local enrichments

A global–local approach to build enrichment functions for the generalized FEM was introduced in [9,10,21]. The approach is based on the solution of interdependent global (structural) and local (crack) scale problems. The local problems focus on the resolution of fine scale features of the solution in the vicinity of three-dimensional cracks while the global problem addresses the macro-scale structural behavior. We denote this class of methods as a *GFEM* with global–local enrichment functions (*GFEM*$^{\mathrm{gl}}$). In this section, we formulate the *GFEM*$^{\mathrm{gl}}$ presented in [21] such that the bulk of the computations can be efficiently done in parallel. In the approach presented below, a local problem is defined for each node of the global mesh whose shape function support intersects a region of interest—like a neighborhood of a crack. It is also conceivable that the region of interest be the entire global domain.

3.1 Formulation of coarse-scale global problem

Consider the domain $\bar{\Omega} = \Omega \cup \partial\Omega \subset \mathbb{R}^3$. The boundary is decomposed as $\partial\Omega = \partial\Omega^u \cup \partial\Omega^\sigma$ with $\partial\Omega^u \cap \partial\Omega^\sigma = \emptyset$.

The strong form of the equilibrium and constitutive equations are given by

$$\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{0} \quad \boldsymbol{\sigma} = \boldsymbol{C} : \boldsymbol{\varepsilon} \quad \text{in } \Omega, \tag{3}$$

where $\boldsymbol{C}$ is Hooke's tensor. The following boundary conditions are prescribed on $\partial\Omega$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \quad \text{on } \partial\Omega^u \quad \boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \quad \text{on } \partial\Omega^\sigma, \tag{4}$$

where $\boldsymbol{n}$ is the outward unit normal vector to $\partial\Omega^\sigma$ and $\bar{\boldsymbol{t}}$ and $\bar{\boldsymbol{u}}$ are prescribed tractions and displacements, respectively.

Let $\boldsymbol{u}_G^0$ denote the generalized or standard FEM solution of the problem defined by (3), (4). This is hereafter denoted as the *initial global problem*. The approximation $\boldsymbol{u}_G^0$ is the solution of the following problem:

Find $\boldsymbol{u}_G^0 \in \boldsymbol{X}_G^0(\Omega) \subset H^1(\Omega)$ such that, $\forall\, \boldsymbol{v}_G^0 \in \boldsymbol{X}_G^0(\Omega)$

$$\begin{aligned}&\int_\Omega \boldsymbol{\sigma}\left(\boldsymbol{u}_G^0\right) : \boldsymbol{\varepsilon}\left(\boldsymbol{v}_G^0\right) \mathrm{d}\boldsymbol{x} + \eta \int_{\partial\Omega^u} \boldsymbol{u}_G^0 \cdot \boldsymbol{v}_G^0 \,\mathrm{d}\boldsymbol{s} \\ &\quad = \int_{\partial\Omega^\sigma} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}_G^0 \,\mathrm{d}\boldsymbol{s} + \eta \int_{\partial\Omega^u} \bar{\boldsymbol{u}} \cdot \boldsymbol{v}_G^0 \,\mathrm{d}\boldsymbol{s}\end{aligned} \tag{5}$$

where, $\boldsymbol{X}_G^0(\Omega)$ is a discretization of $H^1(\Omega)$, a Hilbert space defined on Ω, built with generalized or standard FEM shape functions. In this paper, the *GFEM* is used and the space $\boldsymbol{X}_G^0(\Omega)$ is given by (1). The enrichment functions $L_{\alpha i}$, $\alpha = 1, \ldots, N$, $i = 1, \ldots, D_L$, are taken as polynomials of degree less than or equal to $p-1$. Details can be found, for example, in Sect. 3.2 of [31]. Space $\boldsymbol{X}_G^0(\Omega)$ can also be defined using standard polynomial *FEM* shape functions since cracks are *not* discretized in the initial global problem.

The parameter η in (5) is a penalty parameter determined based on the Young's modulus E and the Jacobian of elements with a face on $\partial\Omega^u$. Other methods to impose Dirichlet boundary conditions can be used as well.

The mesh used to solve problem (5) is typically a coarse quasi-uniform mesh, *regardless of the presence of cracks in the domain*. Figure 2 illustrates one example of such discretization. This mesh and the solution $\boldsymbol{u}_G^0$ are often available from the initial design stage of a structure or mechanical component.

3.2 Formulation of local problems

Let $\Omega_{gl} \subset \Omega$ denote a region of interest like the shaded neighborhood of the crack front shown in Fig. 2. We assume that

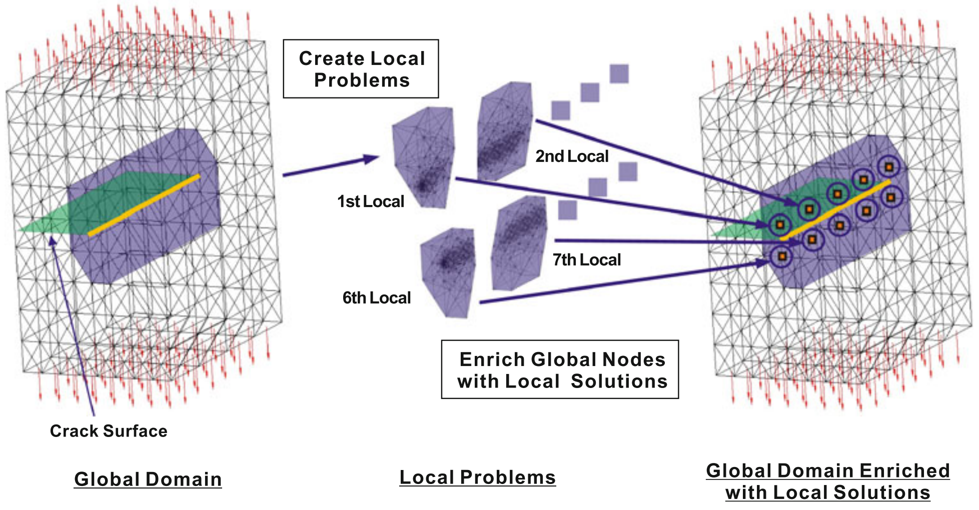


Fig. 2 Illustration of parallel computation of global–local enrichment functions using a cracked bar under tension. Several local problems used for the computation of global–local enrichments are created around the crack front. Each local problem can be sent to a different processor and efficiently solved in parallel. *The crack is shown in the global domain for illustration purposes only*. In the *GFEM*^{gl}, cracks are *not* discretized in the global problem. Instead, global–local enrichment functions are used

Ω_{gl} is the union of clouds ω_α from a global mesh covering Ω. Thus

$$\Omega_{gl} = \bigcup_{\alpha \in \mathscr{I}_{gl}} \omega_\alpha$$

where $\mathscr{I}_{gl}$ denotes an index set of clouds from the global mesh. The parallel *GFEM*^{gl} involves the solution of local boundary value problems defined on clouds ω_α, $\alpha \in \mathscr{I}_{gl}$, of the macro-scale (global) mesh. Each cloud ω_α, $\alpha \in \mathscr{I}_{gl}$, is taken as the domain Ω_L^α of a local boundary value problem. It is also conceivable to use more than one global cloud to define the local domains or to take $\Omega_{gl} = \Omega$. In this paper, we take $\Omega_L^\alpha = \omega_\alpha$. Thus, for each crack, a large number of local problems are defined. Figure 2 illustrates the neighborhood Ω_{gl} of a three-dimensional crack front and local domains Ω_L^α, $\alpha \in \mathscr{I}_{gl}$.

Having the solution of the initial global problem, $\boldsymbol{u}_G^0$ computed as described in the previous section, the following local problems are solved in parallel for all $\alpha \in \mathscr{I}_{gl}$:

Find $\boldsymbol{u}_L^\alpha \in \boldsymbol{X}_L^{hp}(\Omega_L^\alpha) \subset H^1(\Omega_L^\alpha)$ such that, $\forall\ \boldsymbol{v}_L^\alpha \in \boldsymbol{X}_L^{hp}(\Omega_L^\alpha)$

$$\begin{aligned}&\int_{\Omega_L^\alpha} \boldsymbol{\sigma}\left(\boldsymbol{u}_L^\alpha\right) : \boldsymbol{\varepsilon}\left(\boldsymbol{v}_L^\alpha\right) \mathrm{d}\boldsymbol{x} + \eta \int_{\partial\Omega_L^\alpha \cap \partial\Omega^u} \boldsymbol{u}_L^\alpha \cdot \boldsymbol{v}_L^\alpha \,\mathrm{d}s \\ &+\kappa \int_{\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)} \boldsymbol{u}_L^\alpha \cdot \boldsymbol{v}_L^\alpha \,\mathrm{d}s = \int_{\partial\Omega_L^\alpha \cap \partial\Omega^\sigma} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}_L \,\mathrm{d}s \\ &+\eta \int_{\partial\Omega_L^\alpha \cap \partial\Omega^u} \bar{\boldsymbol{u}} \cdot \boldsymbol{v}_L^\alpha \,\mathrm{d}s + \int_{\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)} \left(\boldsymbol{t}\left(\boldsymbol{u}_G^0\right) + \kappa \boldsymbol{u}_G^0\right) \cdot \boldsymbol{v}_L^\alpha \,\mathrm{d}s \qquad (6)\end{aligned}$$

where $\boldsymbol{X}_L^{hp}(\Omega_L^\alpha)$ is a discretization of $H^1(\Omega_L^\alpha)$ defined using the *GFEM* shape functions presented in [31,32]. Details can be found in Section 3.2 of [21].

A key aspect of problem (6) is the use of the coarse-scale solution to compute the boundary condition prescribed on $\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)$. Exact boundary conditions are prescribed on portions of $\partial\Omega_L^\alpha$ that intersect either $\partial\Omega^u$ or $\partial\Omega^\sigma$. The traction vector, $\boldsymbol{t}(\boldsymbol{u}_G^0)$, that appears in the integral over $\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)$ is computed from the coarse-scale solution using Cauchy's relation, i.e.,

$$\boldsymbol{t}\left(\boldsymbol{u}_G^0\right) = \boldsymbol{n} \cdot \boldsymbol{\sigma}\left(\boldsymbol{u}_G^0\right) = \boldsymbol{n} \cdot \left(\boldsymbol{C} : \boldsymbol{\varepsilon}\left(\boldsymbol{u}_G^0\right)\right) \qquad (7)$$

with $\boldsymbol{n}$ the outward unit normal vector to $\partial\Omega_L^\alpha$. The parameter κ is a spring stiffness defined on $\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)$. If the spring stiffness κ is zero or equal to a large (penalty) value, the boundary condition on $\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)$ becomes a Neumann or a Dirichlet boundary condition, respectively. Intermediate values lead to a spring boundary condition. Our numerical experiments presented in [21] show that any value of κ comparable to or larger than the stiffness of the coarse global mesh near $\partial\Omega_L^\alpha \backslash (\partial\Omega_L^\alpha \cap \partial\Omega)$ is acceptable and provides global-local enrichment functions with good approximation properties. For three-dimensional linear elasticity problems, the following spring stiffness κ is recommended [21]:

$$\kappa = \frac{E}{\sqrt[n]{V_0 J}} \qquad (8)$$

where E is the Young's modulus, n is the number of spacial dimensions of the problem, V_0 is the volume of the master element used and J is the Jacobian of the global element across the local boundary where the spring boundary condition is imposed. For further details, refer to [21].

The local problems defined on Ω_L^α, $\alpha \in \mathscr{I}_{gl}$, can be efficiently solved in parallel, since *no communication among processors solving different local problems is required*. This feature of the method is discussed in details in Sect. 4 and numerical evidence is presented in Sect. 5.

3.3 Global–local enrichment functions and enriched global problem

The solutions $\boldsymbol{u}_L^\alpha$ of the local problems on Ω_L^α, $\alpha \in \mathscr{I}_{gl}$, can be used to build generalized FEM shape functions for the coarse global mesh. Equation (2) is used with the partition of unity function, φ_α, provided by the coarse global FE mesh and the enrichment function given by $\boldsymbol{u}_L^\alpha$, i.e.,

$$\boldsymbol{\phi}_\alpha(\boldsymbol{x}) = \varphi_\alpha(\boldsymbol{x})\boldsymbol{u}_L^\alpha(\boldsymbol{x}) \quad \text{(no summation on } \alpha) \tag{9}$$

The local solutions $\boldsymbol{u}_L^\alpha$, $\alpha \in \mathscr{I}_{gl}$, have the role of basis functions for the cloud spaces $\chi_\alpha(\omega_\alpha)$, $\alpha \in \mathscr{I}_{gl}$. Hereafter, $\boldsymbol{u}_L^\alpha$ is denoted a *global–local enrichment function* and the global problem enriched with these functions is denoted an *enriched global problem*. The formulation of this problem is given by
Find $\boldsymbol{u}_G^E \in \boldsymbol{X}_G^E(\Omega) \subset H^1(\Omega)$ such that, $\forall\ \boldsymbol{v}_G^E \in \boldsymbol{X}_G^E(\Omega)$

$$\int_\Omega \boldsymbol{\sigma}\left(\boldsymbol{u}_G^E\right) : \boldsymbol{\varepsilon}\left(\boldsymbol{v}_G^E\right) \mathrm{d}\boldsymbol{x} + \eta \int_{\partial\Omega^u} \boldsymbol{u}_G^E \cdot \boldsymbol{v}_G^E \,\mathrm{d}\boldsymbol{s} = \int_{\partial\Omega^\sigma} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}_G^E \,\mathrm{d}\boldsymbol{s} + \eta \int_{\partial\Omega^u} \bar{\boldsymbol{u}} \cdot \boldsymbol{v}_G^E \,\mathrm{d}\boldsymbol{s} \tag{10}$$

where, $\boldsymbol{X}_G^E(\Omega)$ is the space $\boldsymbol{X}_G^0(\Omega)$ defined in (1) augmented with *GFEM* functions (9). In the case of three-dimensional elasticity problems, global–local enrichments add only three degrees of freedom to nodes $\alpha \in \mathscr{I}_{gl}$ of the coarse global mesh—*the number of enrichment functions per global node does not depend on the number of degrees of freedom of the local problems* (several thousands in general). Furthermore, the hierarchical nature of global–local enrichments implies that the stiffness matrix of the initial global problem is nested in the one of the enriched global problem. Due to these features of the method, the enriched global problem can be efficiently solved using the static condensation scheme introduced in Section A.2 of [9]. In most practical engineering applications, the computational cost to solve the enriched global problem with this strategy is small as demonstrated in [21]. Therefore, in this paper, we focus on the efficiency and accuracy of the parallel solution of the local problems. The reader may also refer to Section 3.3 of [21] for further discussion on space $\boldsymbol{X}_G^E(\Omega)$. Figure 2 illustrates the enrichment of a global mesh with the solution of local problems defined in a neighborhood of a three-dimensional crack.

4 Parallel computation of global–local enrichment functions

In this section, some technical issues related to the parallel implementation of local problem computations are discussed. Hereafter, the methodology described in the previous sections and its parallel implementation are referred to the parallel *GFEM*$^{\mathrm{gl}}$.

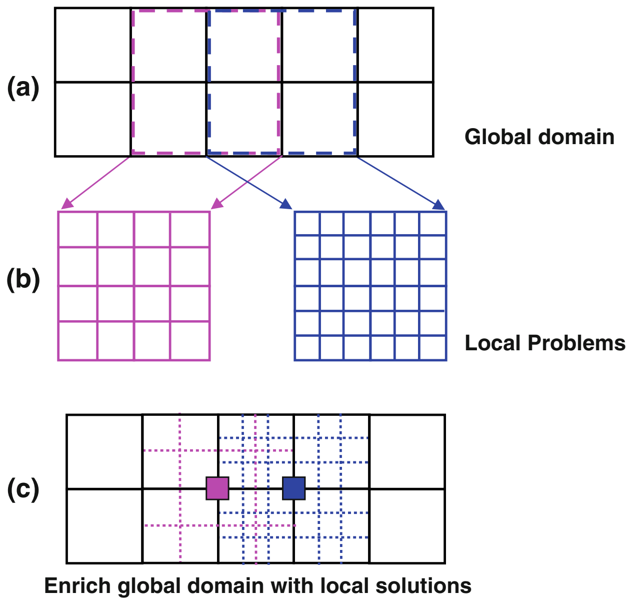


Fig. 3 Incompatibility of local meshes nested in global elements enriched with distinct local solutions. **a** Two clouds used to define small local problems. **b** Local discretizations after refinement. **c** Enrichment of global elements with solutions from the two distinct local problems

4.1 Master-sub local problem approach

The key idea of the parallel *GFEM*$^{\mathrm{gl}}$ is to define a large number of small local problems on a region of interest $\Omega_{gl} \subset \Omega$ as described in Sect. 3.2. The smallest possible local subdomain Ω_L^α corresponds to a single cloud ω_α in the global domain Ω. We adopt these local domains in the parallel implementation described in this paper.

Figure 3a shows two clouds represented by dashed lines in a simple two-dimensional global mesh. Each cloud is used to define a local problem as illustrated in Fig. 3b. The solution of these problems are, in turn, used to enrich the corresponding global clouds. The main technical difficulty with the procedure illustrated in Fig. 3 is how to handle the numerical integration over global elements enriched with distinct local solutions, i.e., elements belonging to the intersection of two or more local domains. If all nodes of a global element are enriched with the same local solution, the numerical integration can be performed with the aid of the local elements nested in the global element. This approach was proposed in [9] and is illustrated in Fig. 4. This procedure, however, cannot be used on global elements enriched with solutions from distinct local problems if the local meshes are not compatible at the intersection of local domains (cf. Fig. 3c). Neither of the two local meshes shown in Fig. 3b is, in general, adequate for numerical integration.

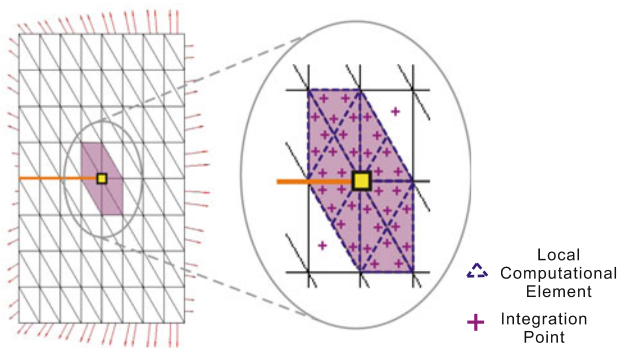


Fig. 4 Illustration of numerical integration scheme in the global elements enriched with local solutions [9]. *Dashed lines* in the figure represent local problem elements nested in global elements. These elements are used to define quadrature points in elements enriched with local solution. *Crosses* represent quadrature points

We address the above issue using the concepts of *master* and *sub-local* problems as illustrated in Fig. 5. Instead of directly creating local problem discretizations from the global mesh, we create first a *master-local* domain extracted from the coarse global mesh. This domain can be *hp*-adapted as if a single local problem is created around a crack. Next, clouds from the coarse global mesh are used to extract *sub-local* domains from the *hp*-adapted master-local domain. No further refinement is done on the sub-local domains after their creation. The sub-local problems defined on sub-local domains can be solved in parallel and their solutions used to enrich global elements. Each cloud from the global mesh contained in the master local domain defines a single local problem in our current implementation. Thus, a large number of small sub-local problems is created. This, as demonstrated in subsequent sections, is beneficial for balancing the workload among processors. Incompatibility between sub-local meshes no longer exists. Therefore, the integration procedure illustrated in Fig. 4 can again be used on global elements.

In some applications, sub-local domains may be created around a region where the solution has strong gradients, is highly oscillatory or is singular, such as in the case of fracture mechanics problems. The quality of the initial global solution is in general poor in those regions. As a result, some sub-local problems may be subjected to poor boundary conditions which may result in global–local enrichment functions with poor approximation properties. This problem can be addressed by performing additional global–local cycles as proposed in [28]. The solution of the enriched global problem is used as boundary conditions for the local problems and the process proceeds as before—solve the local problems and enrich the global discretization with local solutions. One important feature of this approach is that we can obtain improved enriched global solutions with a small amount of additional computations. Since only the boundary conditions provided by the global problem are changed in the sub-local problems [cf. Eq. (6)], the stiffness matrix of each sub-local problem factorized in the first global–local cycle can be reused. Therefore, it is only necessary to rebuild the right-hand side and perform backward and forward substitutions in each sub-local problem. Furthermore, since shape functions built with global-local enrichments are hierarchically added to the global problem, the solution of the enriched global

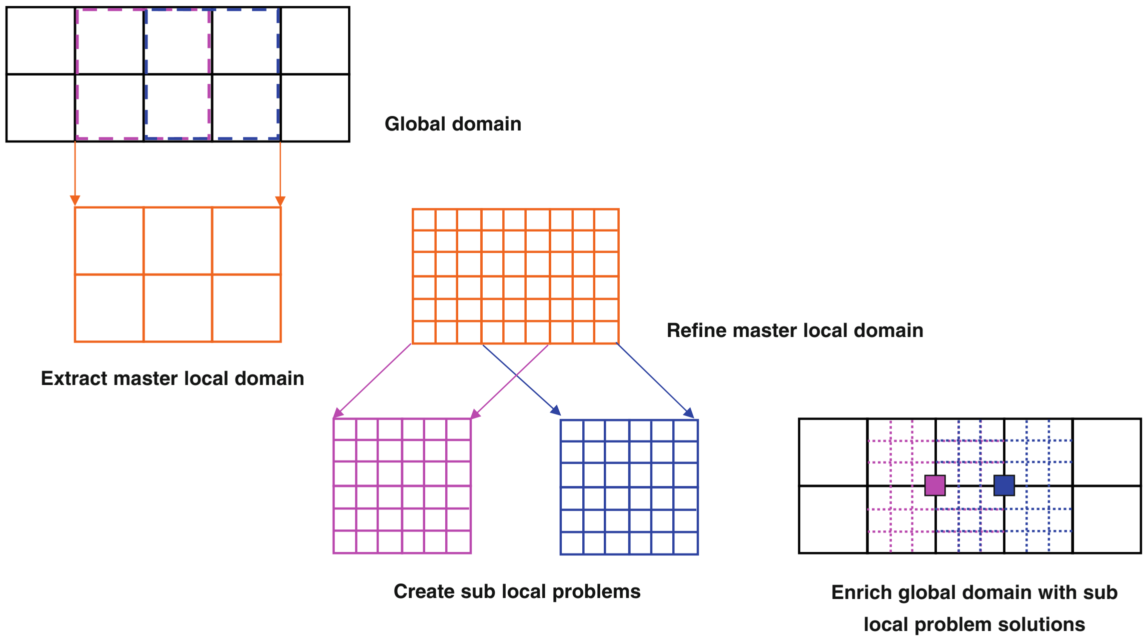

Global domain
Extract master local domain
Refine master local domain
Create sub local problems
Enrich global domain with sub local problem solutions

Fig. 5 Master-sub local problem approach. Sub-local domains are created from a master local domain. This strategy leads to sub-local problems that have compatible meshes at the intersection of their domains

```
Create sub-local problems using the master-sub local problem approach and store them in a list;
Sort sub-local problems in the list;
Activate $N_p$ processors;
foreach processor $i$, $i = 1, \ldots, N_p$ do
    while not all of the sub-local problems in the list are solved do
        while the OpenMP lock is not available do
            Wait until the OpenMP lock becomes available;
        endw
        set an OpenMP lock;
        select the first sub-local problem in the list and remove it from the list;
        unset the OpenMP lock;
        solve the selected sub-local problem;
    endw
endfch
```

Fig. 6 Parallel algorithm for the solution of sub-local problems implemented using OpenMP. A critical region can be set using the OpenMP lock and only one processor is allowed to enter into the critical region

problem can be obtained at a low cost using the algorithm introduced in Section A.2 of [9]. Examples of the application of this approach are presented in Sects. 5.1 and 5.2.

4.2 Parallelization issues

This section focuses on the implementation and load balancing issues of the parallel *GFEM*$^{\mathrm{gl}}$. A simple sorting technique is proposed to improve the parallel efficiency of the algorithm. We also provide an estimate of the largest number of CPUs at which a near uniform distribution of workload among processors can be maintained.

4.2.1 Parallel implementation algorithm

A key advantage of the parallel *GFEM*$^{\mathrm{gl}}$ is that it does not require any communication among processors, and only a pair of scatter-gather communications is involved. All information needed to solve each sub-local problem is obtained from the initial global problem before starting the parallel analysis (a scatter communication). After the parallel solution of each sub-local problem, their solutions are sent back to the global problem for enrichment (a gather communication). This feature of the method greatly simplifies its parallel implementation.

The open multi-processing (OpenMP) programming model [30] is used in our implementation. OpenMP is suitable to run parallel jobs on shared-memory multi-processor platforms. The implementation of the parallel *GFEM*$^{\mathrm{gl}}$ using OpenMP can be summarized as follows:

(i) Create sub-local problems using the master-sub local problem approach discussed in Sect. 4.1 and store them in a list.
(ii) Sort the sub-local problems in the list based on estimates of their computational workload. Sorting criteria are discussed in Sect. 4.2.2.
(iii) Activate N_p processors (user-defined value).
(iv) Select the first sub-local problem in the list and remove it from the list. This operation must be done in a critical region to avoid the so-called *race condition*. Only one processor is allowed to enter into the critical region.
(v) Solve the selected sub-local problem in parallel. Most of CPU time is spent in this step.
(vi) If not all sub-local problems in the list have been solved, go to step (iv) and repeat the procedure. Otherwise, finish the parallel solution of sub-local problems.

High parallel efficiency can be achieved with this algorithm because no processor becomes idle until all sub-local problems in the list are solved. The critical region set up in this algorithm may slightly deteriorate the parallel efficiency, but it is almost negligible when a sufficient large number of local problems is solved in parallel as demonstrated later. The parallelization scheme described above is illustrated in Fig. 6.

In the numerical experiments presented in this paper, the parallelization of the code is done on a SGI Altix 3700 with 1024 Intel Itanium 2 processors located at NCSA (National Center for Supercomputing Applications), Urbana, Illinois, U.S. This machine adopts the Linux operating system by default and was ranked 489th in the top 500 supercomputer list released in November of 2007 [37].

4.2.2 Sorting of sub-local problems

The computational cost to solve each sub-local problem may be significantly different if local mesh refinement or p-enrichment is performed. For example, in fracture problems, a high-level of mesh refinement and enrichment with Westergaard singular functions are performed in elements around the crack front to accurately approximate the solution. As a result, sub-local problems intersecting the crack front have a larger number of degrees of freedom and/or more integration points than sub-local problems far from the crack front [31]. This may cause load unbalance among processors even with the parallel algorithm introduced in Sect. 4.2.1 and result in low parallel efficiency. We observe that a similar difficulty exists when partitioning a domain with adaptive mesh refinement and enrichment in the standard *FEM* [26].

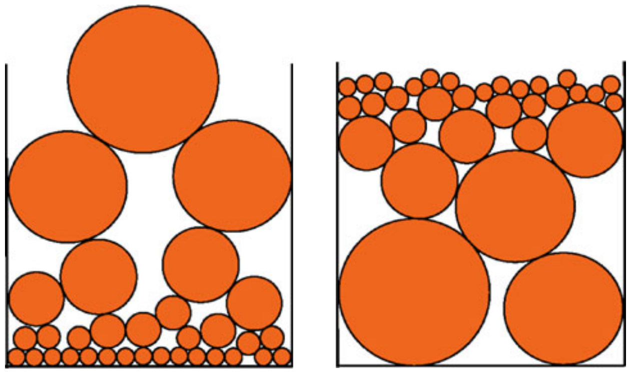

(a) Sorting in ascending order **(b)** Sorting in descending order

Fig. 7 Analogy to sorting of sub-local problems in descending and ascending order

This load unbalance issue can be addressed by sorting the sub-local problems in the list used in the algorithm shown in Fig. 6 based on estimates of their computational load. Figure 7a illustrates the worst possible distribution of sub-local problems in the list. In the figure, sub-local problems are represented by circles and their computational load is represented by the size of the circles. The sub-local problems are sorted in ascending order of their workload—the smallest problem is solved first and the largest one last, thus load unbalance among processors exists if the difference in workload among sub-local problems is significant. Figure 7b represents the opposite case. The sub-local problems are sorted in descending order—the largest problem is solved first and the smallest one last, thus load unbalance among processors can be minimized. If no sorting is done, the parallel efficiency will be somewhere between these two extremes. The parallel performance of these sorting criteria is compared in Sect. 5.2.

4.2.3 Load balance limit

The sorting of sub-local problems in descending order is performed to evenly distribute the workload among processors and improve parallel efficiency. However, even if this sorting criterion is used, there is a limit in the number of CPUs at which load balance among processors can be maintained. This occurs when the computational load of the sub-local problems is significantly different. If the average workload assigned to each processor is smaller than the computational load to solve the largest sub-local problem, load balance among processors cannot be maintained. Therefore, the condition for load balance among processors can be stated by the following inequality

$$\frac{\sum_{i=1}^{N_s} Q_i}{N_p} \geq Q_{\max}, \tag{11}$$

where N_s denotes the number of sub-local problems created, N_p the number of processors provided, Q_i an estimate of the computational cost to solve the ith sub-local problem and $Q_{\max} = \max(Q_1, Q_2, \ldots, Q_{N_s})$. The left and right hand sides of Eq. (11) represent the average workload assigned to each processor and the cost to solve the largest sub-local problem created, respectively.

The largest number of CPUs that satisfies inequality (11), $N^p_{\max}$, is given by

$$N^p_{\max} = \frac{\sum_{i=1}^{N_s} Q_i}{Q_{\max}}. \tag{12}$$

This equation provides an estimate for the largest number of CPUs where load balance among processors can be achieved by assuming that a sufficiently large number of sub-local problems are created.

In addition to the difference in workload among sub-local problems, there are other factors such as initial multi-processor activation cost, processor reactivation cost and cache effect, which may also affect parallel efficiency. Therefore, it is recommended that Eq. (12) be used as a guide to select the number of processors used for a parallel analysis. Another challenge in the application of this equation is how to estimate the computational load to solve each sub-local problem (Q_i) before performing the parallel analysis. According to our numerical experience, the number of floating point operations required to factorize the stiffness matrix of a sub-local problem is a reasonable estimate of the cost to solve a sub-local problem since in general the factorization of the stiffness matrix is the most time-consuming procedure in a finite element analysis. This information is provided by most sparse direct solvers currently available before starting the factorization of a matrix. The validity of Eq. (12) is evaluated in the numerical examples of Sects. 5.2 and 5.3.

5 Numerical experiments

In this section, we analyze several numerical examples to verify the effectiveness and accuracy of the proposed parallel generalized finite element technique.

5.1 Edge cracked bar

As a first example, we analyze a rectangular bar with a through-the-thickness crack with the goal of demonstrating the accuracy of the proposed parallel *GFEM*$^{\mathrm{gl}}$. This example has been analyzed by several researchers [23,34] and thus reliable solutions for the Mode I stress intensity factors, K_I, along the crack front are available. The geometry of the domain and the boundary conditions are illustrated in Fig. 8. The following parameters are used in this example: Poisson's

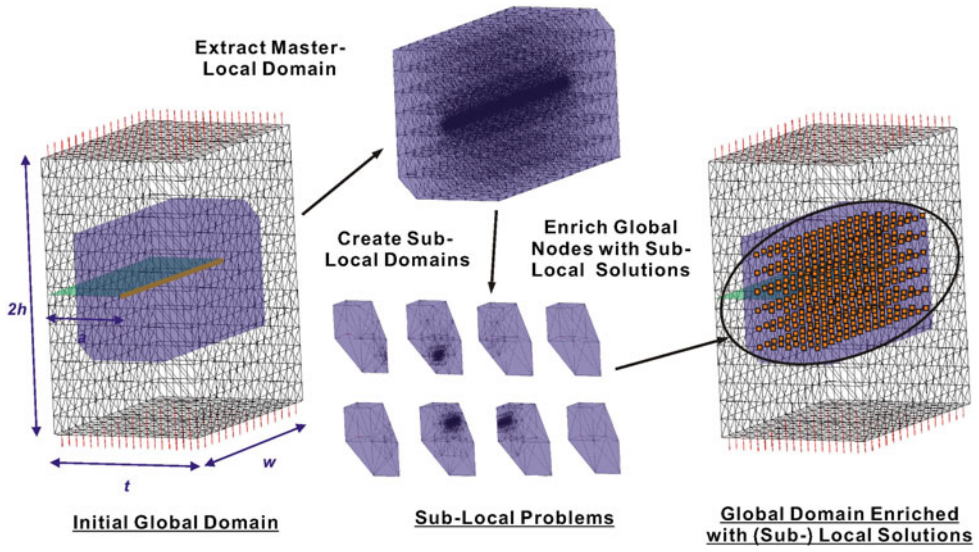

Extract Master-Local Domain
Create Sub-Local Domains
Enrich Global Nodes with Sub-Local Solutions
2h
a
t
w
Initial Global Domain
Sub-Local Problems
Global Domain Enriched with (Sub-) Local Solutions

Fig. 8 Solution of an edge cracked bar using the parallel *GFEM*^{gl} with the master-sub local approach. Only 8 out of 480 sub-local problems are illustrated in the figure

ratio $\nu = 1/3$; Young's modulus $E = 1.0$; Domain dimensions $h/t = 0.875$, $a/t = 0.5$, $w/t = 1.5$. and $t = 2.0$.

Figure 8 illustrates the discretization used in the analysis. The initial global mesh is coarse and is composed of $6 \times (10 \times 15 \times 15)$ tetrahedral elements. Heaviside enrichment functions are used to approximate the solution on elements cut by the crack surface. The crack ends at faces of elements. Westergaard singular enrichment functions are not used in the initial global problem.

The master local domain is created from the coarse global mesh and it has 480 clouds around the crack front. It is *hp*-adapted and enriched with both Heaviside and Westergaard functions. Details on this process can be found in [31, 32]. Next, the master local domain is subdivided into 480 sub-local domains and solved in parallel as illustrated in Fig. 8. Spring boundary conditions provided by the initial global solution are used in each sub-local problem. The spring stiffness is selected using Eq. (8). The mesh of the master local domain is designed such that the ratio of the element size to the characteristic length of the crack (L_e/a) is 0.0192 for the elements intersecting the crack front. The solution of the sub-local problems are used to enrich 480 nodes around the crack front in the global domain. Cubic polynomial shape functions are used in both global and local problems.

5.1.1 Quality of Mode I stress intensity factor

Mode I stress intensity factor (SIF), K_I, extracted along the crack front is normalized using

$$\bar{K}_I = \frac{K_I}{\sigma\sqrt{\pi a}}, \tag{13}$$

where $\bar{K}_I$ denotes the normalized Mode I SIF and σ is the magnitude of the traction applied at the bottom and top surfaces of the rectangular domain.

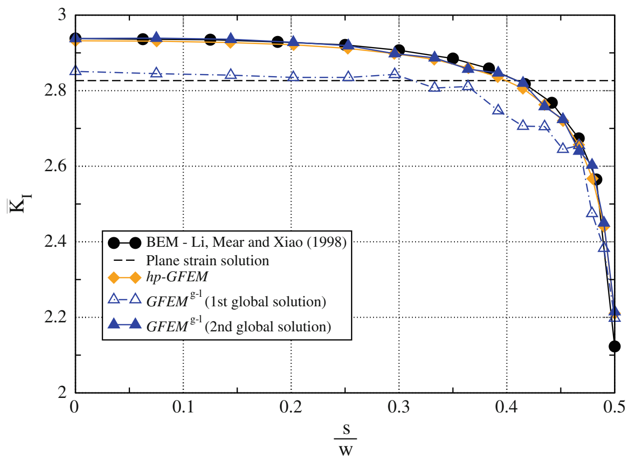


Fig. 9 Mode I SIF extracted from the first and second enriched global solutions

Figure 9 presents the normalized SIF extracted along the crack front using both the *hp-GFEM* for the 3D fractures presented in [31,32] and the proposed parallel *GFEM*^{gl}. The results are plotted in the parametric coordinate system defined on the crack front where $s/w = 0.0$ and $s/w = 0.5$ correspond to the center of the crack front and a crack vertex, respectively. The *hp-GFEM* solution corresponds to the case in which the global mesh is refined and enriched with Heaviside and Westergaard functions as described in [31]. Figure 9 also presents the reference solution provided by Li et al. [23].

The SIF extracted from the (first) global solution enriched with sub-local solutions has an error $e^r(K_I) = 3.01\%$, where $e^r(K_I)$ is a normalized discrete L^2-norm of the difference between the computed SIF and the reference solution defined by

$$e^r(K_I) := \frac{\|e_I\|_{L^2}}{\|\hat{K}_I\|_{L^2}} = \frac{\sqrt{\sum_{j=1}^{N_{ext}} \left(K_I^j - \hat{K}_I^j\right)^2}}{\sqrt{\sum_{j=1}^{N_{ext}} \left(\hat{K}_I^j\right)^2}} \tag{14}$$

where N_{ext} is the number of extraction points along the crack front, $\hat{K}_I^j$ and K_I^j are the reference and computed stress intensity factor values for Mode I at the crack front point j, respectively. This low accuracy happens because the boundary of some sub-local domains intersects the crack front where the quality of the boundary conditions provided by the initial global solution is usually poor. This problem can be addressed by performing an additional global-local cycle as discussed in Sect. 4.1. It can be noted from Fig. 9 that the quality of the second enriched global solution is greatly improved. The relative error in this case is $e^r(K_I) = 1.11\%$.

5.2 Penny shaped crack

The second problem is a circular crack in a cube as illustrated Fig. 10. The objectives of this example are to evaluate the parallel efficiency and accuracy of the proposed parallel *GFEM*$^{\text{gl}}$ with master-sub local problem approach. The effect of the sorting techniques discussed in Sect. 4.2.2 and the validity of Eq. (12) are also investigated.

A tensile traction of magnitude σ is applied in the y-direction at the top and bottom surfaces of the domain. The following parameters are assumed in this problem: Cube dimension $2L = 2.0$; crack radius $a = 0.5$; vertical traction $\sigma = 1.0$; Young's modulus $E = 2.0 \times 10^5$; Poisson's ratio $\nu = 0.3$.

The solution of this problem is computed following the same steps described in Sect. 5.1. The global domain is discretized with a uniform coarse mesh of $6 \times (10 \times 10 \times 10)$ tetrahedral elements as shown in Fig. 10. In contrast with the previous example, the crack is *not* discretized in the initial global problem. A single master-local domain containing the entire circular crack surface is created from the coarse global mesh. It has 495 clouds around the crack surface. Dirichlet boundary conditions provided by the initial global problem are used in each sub-local problem. The master local domain is locally refined around the crack front such that the ratio of element size to characteristic crack length (L_e/a) along the crack front is 0.0280. The master local domain is subdivided into 495 sub-local domains and solved in parallel. Both Heaviside and Westergaard enrichment functions are used in the local problems. The solutions of these sub-local problems are used as enrichment functions in the coarse global domain as illustrated in Fig. 10. Cubic polynomial shape functions are used in both global and local problems.

5.2.1 Parallel performance

In this section, we investigate the parallel performance of the proposed parallel *GFEM*$^{\text{gl}}$ for the penny shaped crack example. The total CPU time required for the parallel solution of all 495 sub-local problems is measured for several number of processors. These results are then used to compute parallel efficiency and speed-up using [33]

$$\text{Parallel speed-up} = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

$$\text{Parallel efficiency} = \frac{\text{Parallel speed-up}}{\text{Number of processors used}}.$$

The two sorting techniques discussed in Sect. 4.2.2 are used and their effects on the parallel efficiency are examined. Equation (12) predicts that load unbalance starts to occur at $N^p_{\max} = 22.6$ if the sorting in descending order is used. The accuracy of this estimate is evaluated by comparing it with the numerical result presented in this section.

Tables 1 and 2 show the total CPU required for the parallel solution of sub-local problems, the parallel efficiency and speed-up with respect to the number of CPUs for the two sorting criteria. The maximum number of CPUs

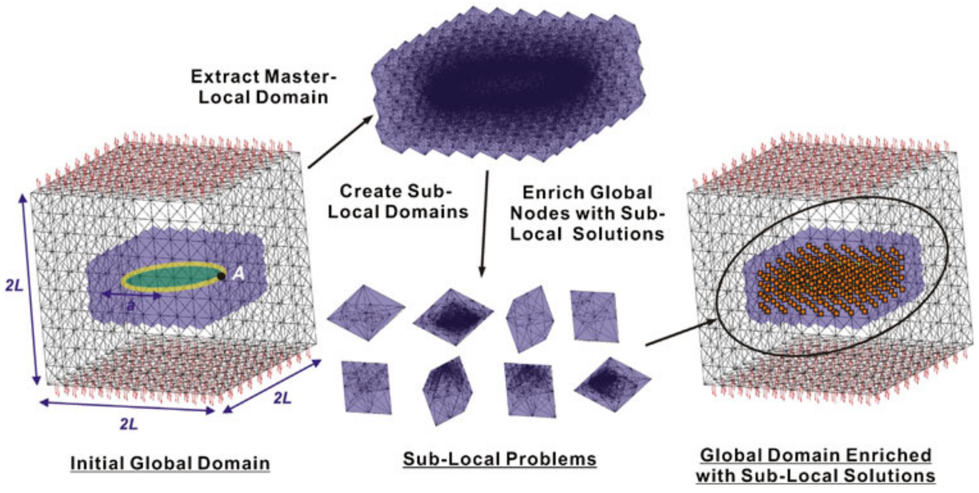

Extract Master-Local Domain
Create Sub-Local Domains
Enrich Global Nodes with Sub-Local Solutions
2L
2L
2L
A
a
Initial Global Domain
Sub-Local Problems
Global Domain Enriched with Sub-Local Solutions

Fig. 10 Solution of a cube with a penny shaped crack using the parallel *GFEM*$^{\text{gl}}$ with master-sub local approach. Only 8 out of 495 sub-local problems are illustrated in the figure

Table 1 Parallel performance for the penny shaped crack problem with sorting of sub-local problems in descending order

Number of processors	CPU time (s)	Parallel efficiency	Speed-up
1	4653.0	N/A	N/A
2	2423.8	0.960	1.920
4	1218.7	0.955	3.818
8	612.5	0.950	7.597
16	309.3	0.940	15.046
20	247.1	0.942	18.834
22	233.1	0.907	19.960
32	219.8	0.662	21.169

Table 2 Parallel performance for the penny shaped crack problem with sorting of sub-local problems in ascending order

Number of processors	CPU time (s)	Parallel efficiency	Speed-up
1	4653.0	N/A	N/A
2	2441.0	0.953	1.906
4	1254.1	0.928	3.710
8	707.0	0.823	6.581
16	384.4	0.757	12.105
20	325.2	0.715	14.308
22	313.7	0.674	14.834
32	282.2	0.515	16.491

provided for the parallel computation is set to 32 by referring to the estimate given by Eq. (12). It can be observed from the results in the tables that the sorting in descending order delivers much higher parallel efficiency, which is above 90% for 22 or less processors, than the sorting in ascending order. For example, when 22 processors are used, the parallel efficiency obtained by the former is 90.7%, while that obtained by the latter is only 67.4%.

The results in Tables 1 and 2 are plotted in Fig. 11. The horizontal axis represents the total number of processors used and the vertical axes on the left and right side indicate parallel efficiency and speed-up, respectively. The speed-up obtained by the sorting in descending order exhibits almost linear scalability up to $N_p = 22$ and then the slope drops at around $N_p = 22$. The curve obtained by the sorting in ascending order has a similar shape, but it starts deviating from the linear scalability at $N_p = 8$ and its parallel efficiency is much lower.

We display the number of sub-local problems solved and CPU time spent in each processor in Fig. 12 in order to investigate the load balance among processors in the two sorting cases when $N_p = 8$. From this plot, the following observations can be made. First, the sorting in descending order distributes the workload uniformly among processors, while the number of sub-local problems solved in each processor

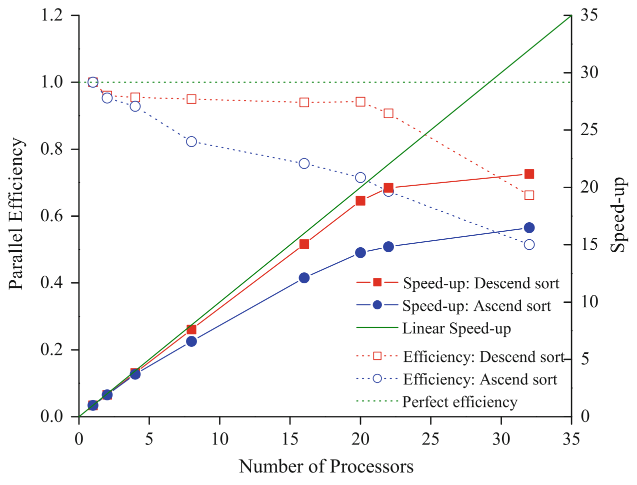


Fig. 11 Parallel performance for the penny shaped crack example

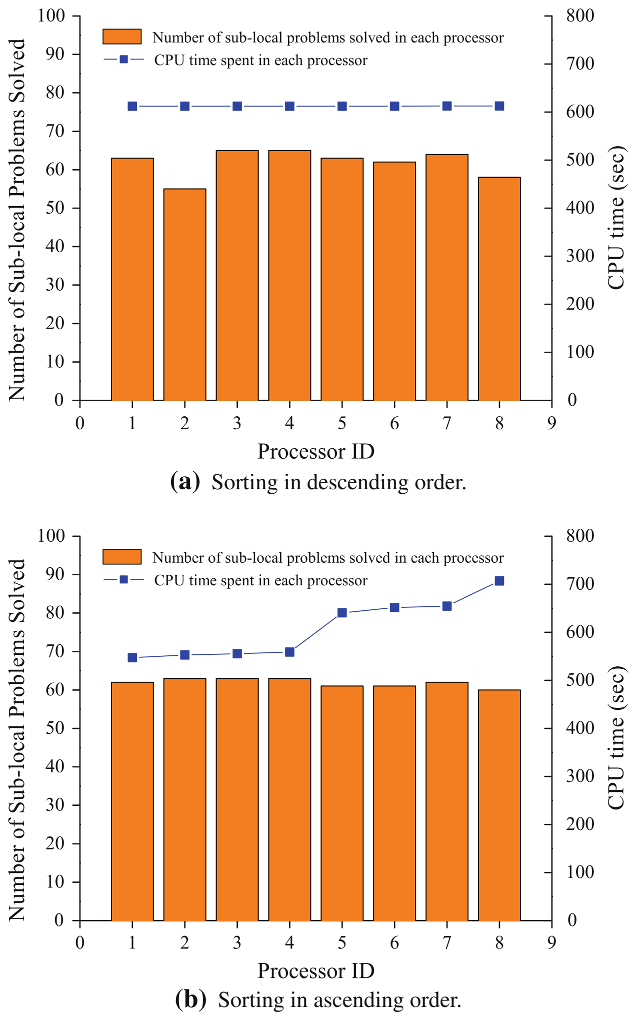
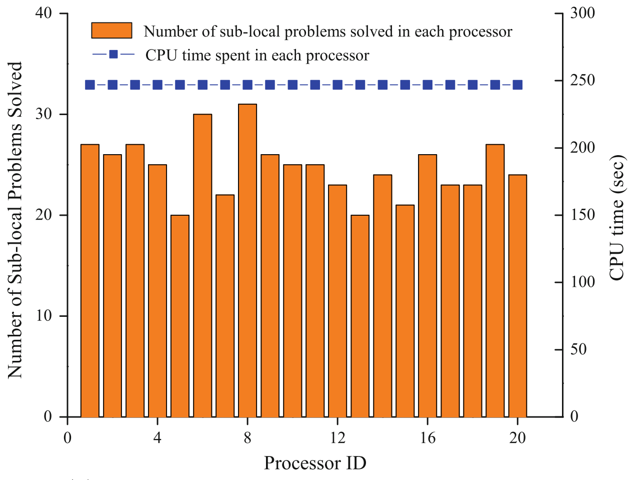
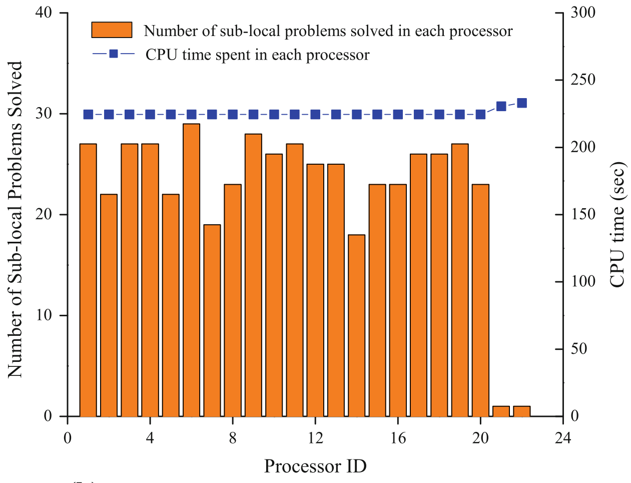


Fig. 12 Effect of the sorting criteria on the load balance among processors for $N_p = 8$

is uneven. As shown in Fig. 12a, the CPU time spent in each processor is almost the same. Second, the sorting in ascending order does exactly the opposite. Almost the same



(a) Workload distribution to each processor for $N_p = 20$.



(b) Workload distribution to each processor for $N_p = 22$.

Fig. 13 Load balance limit for the penny shaped crack example. Sub-local problems are sorted in descending order

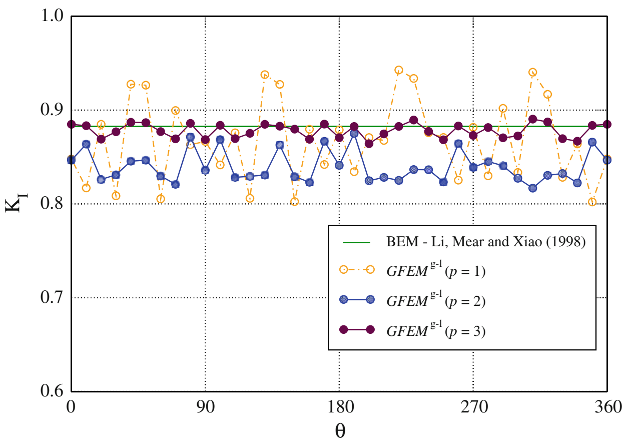


Fig. 14 SIFs extracted along the circular crack front for different order of polynomial shape functions

number of sub-local problems are solved in each processor, but the workload is not uniformly distributed among processors, leading to an uneven amount of CPU time spent in each processor.

Next, we investigate why the parallel efficiency suddenly drops at $N_p = 22$ even with the sorting in descending order. Figure 13 shows the same quantities shown in Fig. 12 but for the cases $N_p = 20$ and $N_p = 22$. These results are obtained by using the sorting in descending order. Figure 13a indicates that the workload is uniformly distributed to each processor when 20 processors are used and the number of sub-local problems solved in each processor is quite uneven as in the case with $N_p = 8$. However, an uniform distribution of the workload can no longer be achieved if 22 processors are used since the cost to solve the largest sub-local problem is larger than the averaged workload assigned to other processors. Figure 13b shows that the computational cost to solve each of the two largest sub-local problems is larger than the averaged workload of other processors. This is precisely the case that Eq. (11) describes. Therefore, if more than 22 CPUs are used, parallel efficiency is deteriorated as already shown in Fig. 11.

The results shown in Fig. 13 confirm that the estimate of the load balance limit provided by Eq. (12) is quite accurate. Although the proposed master-sub local problem approach has a limitation in the number of processors at which load balance can be maintained, the limit is determined by the cost to solve the largest sub-local problem, which is in our experience <5 min because a sub-local domain is defined by a single cloud in the coarse global mesh as discussed in Sect. 4.1. Thus, this limitation is not relevant from a practical point of a view.

The numerical results in this section indicate that the master-sub local problem approach with sorting in descending order can achieve parallel efficiency above 90% up to the load balance limit provided by Eq. (12). This demonstrates a high scalability of the proposed parallel $GFEM^{gl}$ technique which is possible due to its unique feature that does not require any communication among processors during the parallel solution of sub-local problems.

5.2.2 *Quality of extracted stress intensity factors*

Figure 14 presents the Mode I SIF extracted from the parallel $GFEM^{gl}$ solution. A reference SIF value provided by Li et al. [23] is given by,

$$K_I^{\text{ref.}} = \frac{2.213}{\pi}\sigma\sqrt{\pi a} \tag{15}$$

where a denotes the crack radius and σ is the magnitude of the traction applied at the bottom and top surfaces of the rectangular domain. Li et al. [23] report K_I at point A indicated in Fig. 10, which corresponds to $\theta = 0°$ or $360°$ in Fig. 14. Although this value does not represent K_I along the whole crack front due to the finite size of the domain, the exact K_I only slightly varies from it. Therefore, we use it as a reference value for the Mode I SIF along the whole crack front.

Similar to the edge cracked bar example presented in Sect. 5.1, the quality of boundary conditions used on sub-local problems created along the crack front is poor if their boundaries are intersected by the crack front. Therefore, we follow the same procedure used in Sect. 5.1.1 and the SIF shown in Fig. 14 are extracted from the enriched global solution obtained after three global–local cycles. Figure 14 shows the SIFs extracted from the $GFEM^{gl}$ solutions computed with polynomial shape functions of degree $p = 1, 2, 3$. It can be noted from the figure that K_I converges to the reference value as the order of polynomial shape functions increases. The SIF for the case $p = 3$ shows a good agreement with the reference SIF and the magnitude of the oscillations is small along the crack front. The relative error for this case is $e^r(K_I) = 0.05\%$.

5.3 Mechanical manifold with multiple cracks

In this section, we analyze a problem with an industrial level of complexity using the parallel $GFEM^{gl}$ and investigate the performance of the parallel solution of sub-local problems. The problem is a mechanical manifold with multiple cracks, which is a thin tube-like structure. The geometry and boundary conditions are shown in Fig. 15. The Young's modulus and Poisson's ratio are assumed to be $E = 2.0 \times 10^5$ and $\nu = 0.3$, respectively.

The top, front and back views of the manifold are provided in Figs. 15, 16 and 17, respectively. Six cracks are modeled in the regions with stress concentration where they are likely to nucleate and grow under the given geometry and boundary conditions. The first and second cracks shown in Figs. 16 and 17 have two crack fronts due to the geometry of the domain. A master-local domain is created at each crack front. All other cracks, from the third to the sixth one, have only one crack front and one master-local domain is created in the neighborhood of each front. Thus, a total of eight master-local domains are extracted from the global mesh. Each master-sub local domain is locally refined around its crack front and enriched with Heaviside and singular Westergaard functions. Next, master-local domains 1 through 8 are subdivided into 229, 172, 116, 191, 26, 37, 109 and 103 sub-local problems, respectively. The total number of sub-local problems created

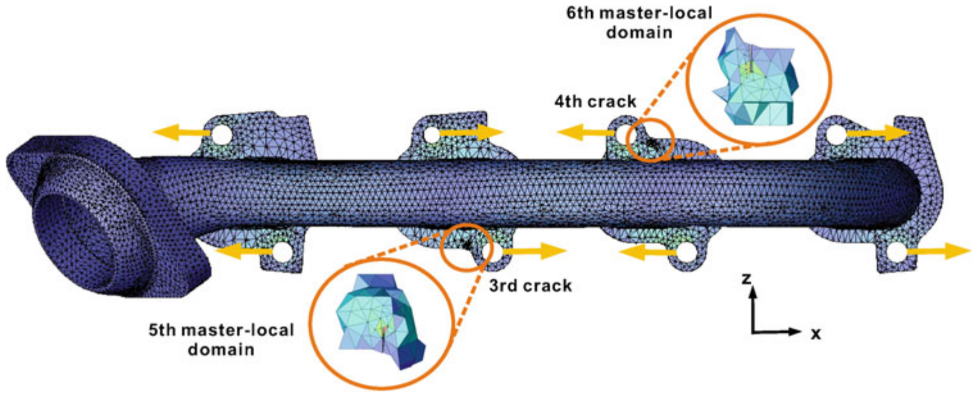


Fig. 15 *Top view* of the mechanical manifold with multiple cracks. The structure is subjected to displacement boundary conditions represented by the *arrows* at the eight holes. The prescribed displacement vectors in the positive and negative x-directions are $(u_x, u_y, u_z) = (0.02, 0, 0)$ and $(u_x, u_y, u_z) = (-0.02, 0, 0)$, respectively. Homogeneous traction boundary conditions are applied elsewhere on the boundary. The 5th and 6th master-local domains extracted around the 3rd and 4th crack surfaces, respectively, are also shown in the figure

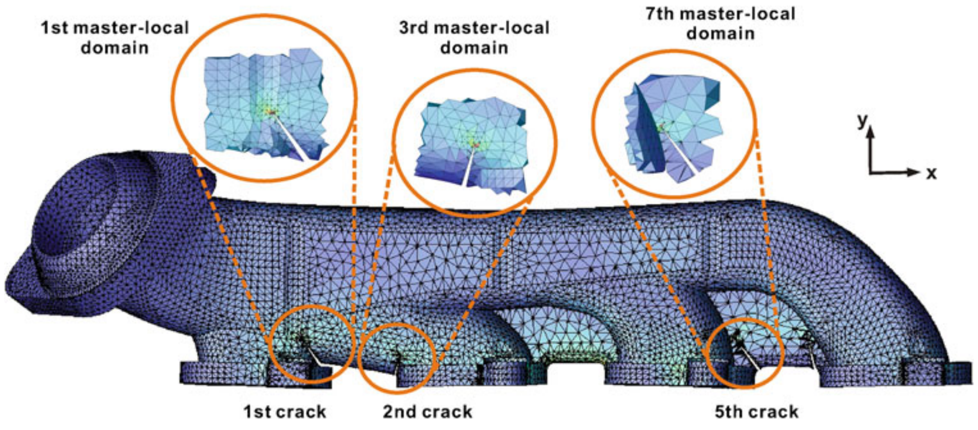


Fig. 16 *Front view* of the mechanical manifold with multiple cracks. The 1st, 3rd and 7th master-local domains extracted around the 1st, 2nd and 5th cracks, respectively, are shown in the figure

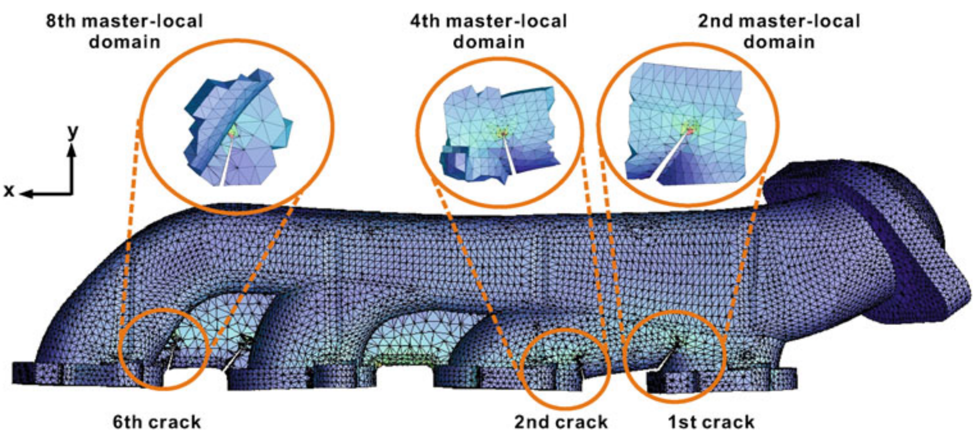


Fig. 17 *Back view* of the mechanical manifold with multiple cracks. The 2nd, 4th and 8th master-local domains extracted around the 1st, 2nd and 6th cracks, respectively, are shown in the figure

Table 3 Parallel performance of a mechanical manifold example with sorting in descending order

Number of processors	CPU time (s)	Parallel efficiency	Speed-up
1	1537.4	N/A	N/A
2	778.1	0.988	1.976
4	392.1	0.980	3.921
8	198.3	0.969	7.752
16	101.1	0.951	15.214
18	92.3	0.926	16.666
20	86.3	0.891	17.823
32	66.4	0.724	23.161

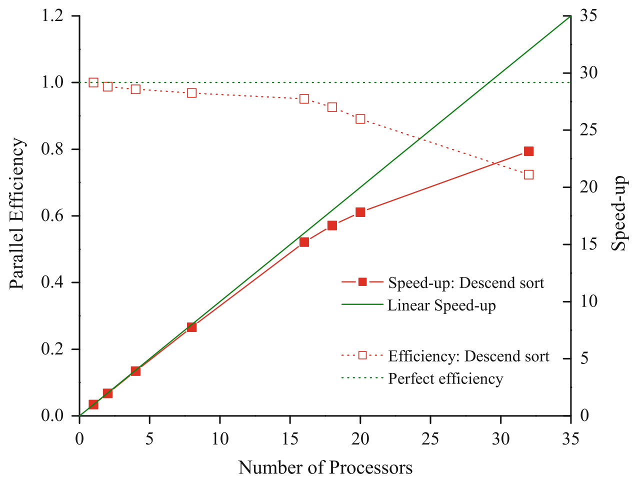


Fig. 18 Parallel performance of the mechanical manifold example

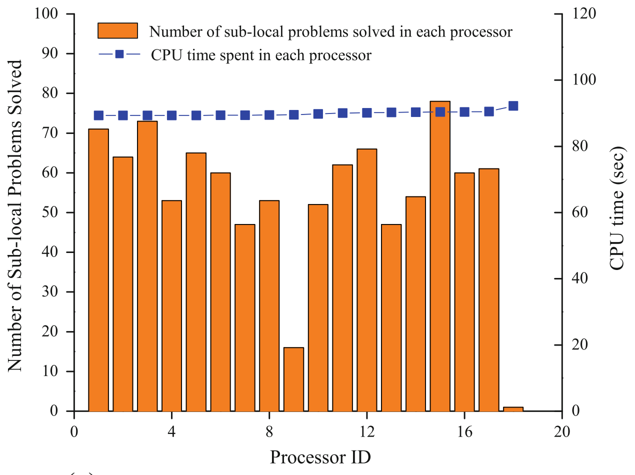
is 983. Quartic polynomial shape functions are used in all sub-local problems. The 983 sub-local problems lead to additional 2,949 degrees of freedom in the enriched global problem. A comparable *hp-GFEM* discretization with the same level of mesh refinement as in the sub-local problems and quartic polynomial shape functions would have 1,605,960 degrees of freedom.

5.3.1 Parallel performance

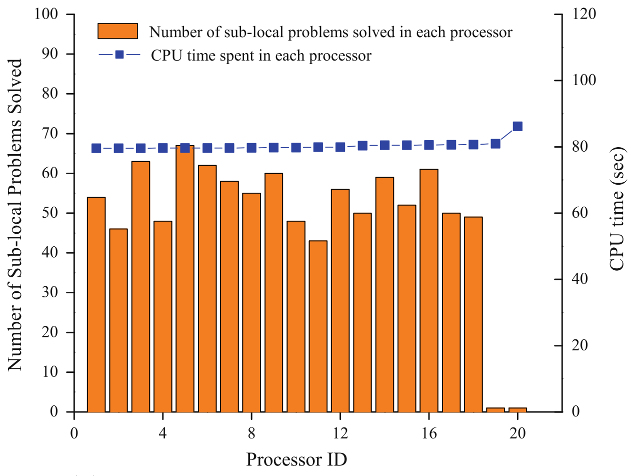
The parallel performance of the sub-local problem computations is examined in this section. The total CPU time required for the parallel solution of all 983 sub-local problems is measured for several number of processors. The parallel efficiency and speed-up are computed from those results. The sorting in descending order is used in all computations. The load balance limit is estimated as $N^p_{\max} = 20.1$ using Eq. (12).

Table 3 lists the CPU time required to solve all sub-local problems in parallel, the efficiency and speed-up with respect to the number of processors. It can be noted from the table that a high parallel efficiency is maintained up to $N_p = 20$ (89.1% when 20 processors are provided). Figure 18 plots the data in Table 3. The horizontal axis represents the total number of processors used and the vertical axes on the left and right side indicate parallel efficiency and speed-up, respectively. Similar to the results in Fig. 11, the speed-up curve shows nearly linear scalability up to $N_p = 18$ and then the slope drops.

Figures 19a and b present the number of sub-local problems solved and the CPU time in each processor for $N_p = 18$ and $N_p = 20$, respectively. Figure 19a indicates that the CPU time spent on the last processor is slightly larger than on the others and only one sub-local problem is solved in that processor when 18 processors are provided. This means that the computational cost to solve the largest sub-local problem is larger than the average workload assigned to other processors and perfect load balance among processors is not maintained. Figure 19b shows a situation where parallel efficiency is slightly more deteriorated and the cost to solve the two largest sub-local problems exceeds the averaged workload per processor. However, there is no significant practical relevance to further reduce the analysis time by providing more processors since the CPU time spent when $N_p = 20$ is just 86.3 s.



(a) Workload distribution for each processor for $N_p = 18$.



(b) Workload distribution for each processor for $N_p = 20$.

Fig. 19 Load balance limitation for the mechanical manifold example

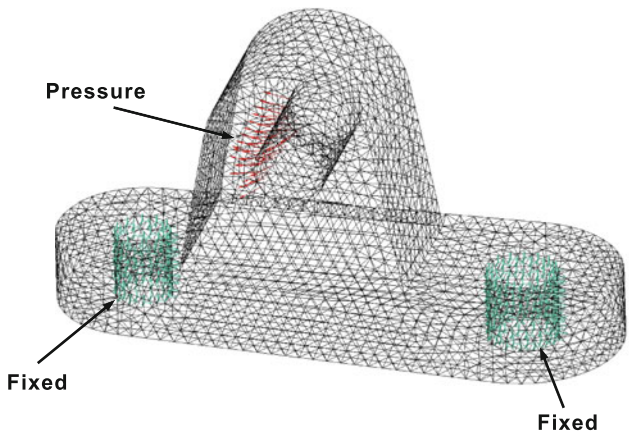


Fig. 20 Boundary conditions and mesh for a structural component

The parallel performance and load balance described above show again a good agreement with the estimate provided by Eq. (12), which is $N^p_{\max} = 20.1$. All of the numerical results in this section demonstrate that high parallel efficiency can be achieved by the proposed parallel *GFEM*$^{\text{gl}}$ with the master-sub local problem approach up to the load balance limit.

5.4 Structural component

The proposed parallel *GFEM*$^{\text{gl}}$ is not restricted to the analysis of three-dimensional cracks. It can be used, for example, to perform aggressive *hp*-extensions in large computational models while adding only three degrees of freedom per node enriched with global–local functions. This capability of the method is demonstrated in this section. The geometry and boundary conditions of the problem are shown in Fig. 20. The global mesh used in this example consists of 3,849 nodes and 15,527 tetrahedral elements. Linear elastic material properties are assumed with Young's modulus $E = 10^5$ and Poisson's ratio $\nu = 0.33$.

Figure 21 illustrates the discretization used for the parallel solution procedure. A single master domain that is equal to the global domain is created. To investigate the effectiveness of the proposed parallel *GFEM*$^{\text{gl}}$ for large problems, different levels of uniform mesh refinement are performed at the master-local domain. A total of 3,849 sub-local problems are extracted from the refined master-local mesh. This is equal to the number of nodes in the coarse global mesh. These problems are solved in parallel and their solutions used to enrich the global problem. Cubic polynomial functions are used in both the global and sub-local problems.

Table 4 lists the number of degrees of freedom (DOFs) required for the parallel *GFEM*$^{\text{gl}}$ analysis when utilizing three levels of uniform mesh refinement (hereafter denoted Levels 1, 2 and 3). The initial global problem has 115,470 DOFs while the enriched global has 115,470 + 3 × (3,849) = 127,017 DOFs, *regardless of the size of the sub-local problems*. The range of sub-local problem sizes is also listed for each level of mesh refinement. The numbers of DOFs that would be required for an *hp-GFEM* analysis of this example are presented in the table. They are obtained by assuming that the same level of mesh refinement and polynomial enrichment used in the master-local domain is directly performed on the global mesh. It can be noted from the table that the *hp-GFEM* analysis with Level 3 mesh refinement requires more then 9 million DOFs, which would be very difficult, if possible at all, to solve on a single processor. In contrast, the largest number of DOFs needed for the parallel *GFEM*$^{\text{gl}}$ analysis is 127,017, which is still solvable on a single processor and the largest sub-local problem created by Level 3 mesh refinement requires around 40,000 DOFs.

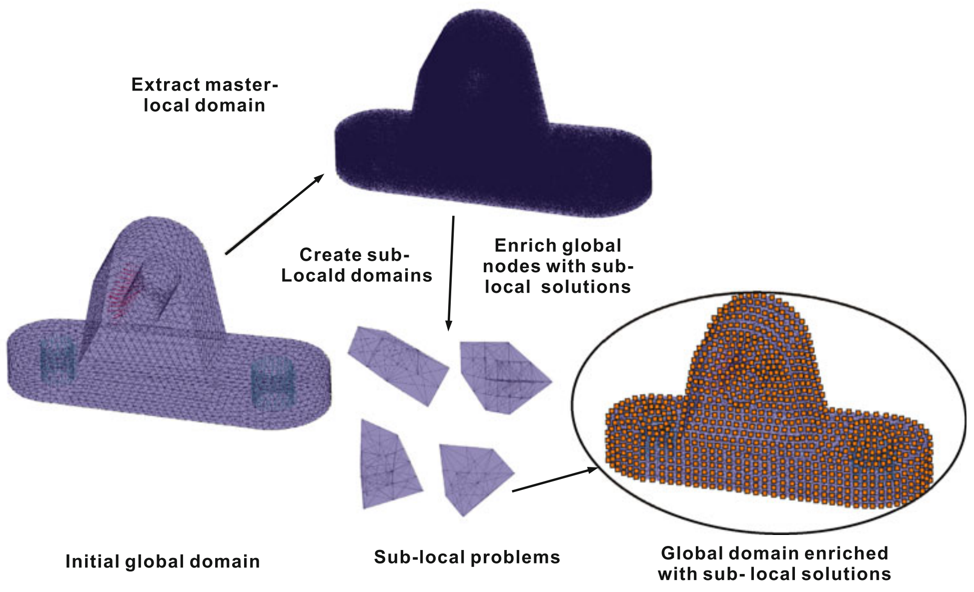

Extract master-local domain
Create sub-Locald domains
Enrich global nodes with sub-local solutions
Initial global domain
Sub-local problems
Global domain enriched with sub- local solutions

Fig. 21 Solution of a structural component using the parallel $GFEM^{gl}$ with the master-sub local approach. The refined master-local domain mesh in the figure corresponds to Level 1 of mesh refinement. Only 4 out of 3,849 sub-local problems are illustrated in the figure

Table 4 Number of degrees of freedom required for *hp-GFEM* and parallel $GFEM^{gl}$ analyses with three different levels of uniform mesh refinement

Level of uniform mesh refinement	*hp-GFEM*	Parallel $GFEM^{gl}$			
		Initial global	Sub-local		Enriched global
			Minimum	Maximum	
1	1,150,590		420	5,580	
2	3,377,550	115,470	720	15,180	127,017
3	9,235,530		1,590	40,170	

Table 5 Parallel performance for three levels of mesh refinement at sub-local problems

Number of processors	CPU time (s)			Parallel efficiency		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
1	3534.3			N/A		
2	1843.6	18951.8		0.959	N/A	
4	938.0	9842.6		0.942	0.963	
8	479.4	5067.5		0.922	0.935	
16	240.6	2829.5	17895.8	0.918	0.837	N/A
32	126.0	1449.1	9780.7	0.877	0.817	0.913
64		553.8	5220.6		1.069	0.856
128			2145.9			1.068

5.4.1 Parallel performance

The performance of the parallel $GFEM^{gl}$ is investigated in this section. The total CPU time to solve in parallel all 3,849 sub-local problems is measured for up to 128 processors. The parallel efficiency is computed for three levels of mesh refinement as shown in Table 5. The computational load and required memory for mesh refinement Levels 2 and 3 are large and more than one processor must be used. The required minimum number of processors is determined for each case. The parallel efficiency is measured with respect to the CPU time obtained with the minimum number of processors, not with respect to the serial solution as was the case in previous examples. Sub-local problems are sorted in descending order.

The load balance limits for mesh refinement Levels 1, 2 and 3 are $N^p_{\max} = 538.7$, $N^p_{\max} = 370.6$ and $N^p_{\max} = 386.7$, respectively, according to the estimate provided by

Equation (12). All of them are well above the maximum number of processors used in our computations (i.e. 128 processors). As expected, from the estimate, the results in Table 5 show that high parallel efficiency is maintained up to the maximum number of processors considered. For instance, the parallel efficiency for mesh refinement Level 3 is about 85% when 64 processors are provided. It can also be observed from the table that superlinear scalability is achieved when 64 and 128 processors for Levels 2 and 3, respectively, are used. This is likely due to cache memory and memory latency effects of the non-uniform memory access (NUMA) system, on which the SGI Altix 3700 is based. Since the problem size is very large for Levels 2 and 3, a processor uses memory from other processors in the NUMA system when a small number of processors are assigned for parallel computations. However, it does not use cache memory from other processors. Thus, when the number of processors performing computations is increased, more cache memory is used, even if the total number of processors locked for the parallel computation remains the same. This can contribute to superlinear scalability. In addition, when a processor uses memory from several other processors, memory access latency increases. Thus, as the number of processors performing computations increases, the latency decreases if the number of processors locked for the parallel computations remains the same. This also contributes to superlinear scalability.

The results in the table confirm that high parallel efficiency can be achieved by the proposed parallel $GFEM^{gl}$ and increasing the number of processors does not result in significant deterioration of efficiency.

6 Summary and concluding remarks

In this paper, we introduced a parallel generalized finite element method that uses customized enrichment functions for applications where limited a priori knowledge about the solution is available. The enrichment functions can be efficiently constructed through parallel computations. The methodology involves the parallel solution of local boundary value problems using boundary conditions from a coarse global problem. The local solutions are in turn used to enrich the global space using the partition of unity of framework. In this procedure, load unbalance is addressed by defining a larger number of small local problems than the number of parallel processors and by sorting and solving the local problems based on estimates of their workload. The compatibility between local meshes in overlapping regions is guaranteed by the master-sub local problem approach presented in Sect. 4.1. A simple and effective estimate of the largest number of processors where load balance among processors is maintained is proposed. The effectiveness of the method was investigated in terms of accuracy and parallel efficiency through several three-dimensional fracture mechanics examples. The main conclusions of this paper are as follows:

- The proposed parallel $GFEM^{gl}$ allows efficient parallel solution of practical engineering problems such as the analysis of three-dimensional cracks that are difficult to solve using available numerical methodologies. The numerical experiments performed in this paper show that parallel efficiency above 80% can be achieved up to the load balance limit;
- The sorting of sub-local problems in descending order improves parallel efficiency significantly and the proposed estimate for the load balance limit expressed by Eq. (12) shows good accuracy;
- The quality of the parallel $GFEM^{gl}$ solutions can be improved by performing additional global–local cycles when sub-local problems are subjected to poor boundary conditions. The solutions computed with the proposed approach are of comparable quality to that of the *hp-GFEM* [31,32].

Acknowledgments The partial support of this work by the National Center for Supercomputing Applications and the University of Illinois at Urbana-Champaign, under the auspices of the NCSA/UIUC Faculty Fellows Program, and by the U.S. Air Force Office of Scientific Research under contract number FA9550-09-1-0401 are gratefully acknowledged.

References

1. Babuška I, Caloz G, Osborn JE (1994) Special finite element methods for a class of second order elliptic problems with rough coefficients. SIAM J Numer Anal 31(4):945–981
2. Babuška I, Melenk JM (1995) The partition of unity finite element method. Technical Report BN-1185, Inst. for Phys. Sc. and Tech., University of Maryland
3. Babuška I, Melenk JM (1997) The partition of unity finite element method. Int J Numer Methods Eng 40:727–758
4. Belytschko T, Black T (1999) Elastic crack growth in finite elements with minimal remeshing. Int J Numer Methods Eng 45:601–620
5. Belytschko T, Gracie R, Ventura G (2009) A review of extended/generalized finite element methods for material modeling. Model Simul Mater Sci Eng 17:24. doi:10.1088/0965-0393/17/4/043001
6. Duarte CA (1996) The hp Cloud Method. PhD dissertation, The University of Texas at Austin, Austin, TX, USA
7. Duarte CA, Babuška I (2005) A global-local approach for the construction of enrichment functions for the generalized fem and its application to propagating three-dimensional cracks. In: Leitão VMA, Alves CJS, Duarte CA (eds) ECCOMAS Themat Conf Meshless Methods, Lisbon, Portugal, pp 11–14, July 2005 (8 pages)
8. Duarte CA, Babuška I, Oden JT (2000) Generalized finite element methods for three dimensional structural mechanics problems. Comput Struct 77:215–232
9. Duarte CA, Kim D-J (2008) Analysis and applications of a generalized finite element method with global–local enrichment functions. Comput Methods Appl Mech Eng 197(6–8):487–504. doi:10.1016/j.cma.2007.08.017

10. Duarte CA, Kim D-J, Babuška I (2007) Chapter: a global–local approach for the construction of enrichment functions for the generalized fem and its application to three-dimensional cracks. In: Leitão VMA, Alves CJS, Duarte CA (eds) Advances in meshfree techniques. Comput Methods Appl Sci, vol 5. Springer, The Netherlands. ISBN: 978-1-4020-6094-6
11. Duarte CAM, Oden JT (1995) Hp clouds—a meshless method to solve boundary-value problems. Technical Report 95-05, TICAM, The University of Texas at Austin
12. Duarte CAM, Oden JT (1996) An *hp* adaptive method using clouds. Comput Methods Appl Mech Eng 139:237–262
13. Duarte CAM, Oden JT (1996) *Hp* clouds – An *hp* meshless method. Numer Methods Partial Differ Equ 12:673–705
14. Duflot M, Bordas S (2008) XFEM and mesh adaptation: a marriage of convenience. In: Eighth World Congr Comput Mech, Venice, Italy, July 2008
15. Farhat C (1988) A simple and efficient automatic fem domain decomposer. Comput Struct 28:579–602
16. Farhat C, Roux FX (1991) A method of finite element tearing and interconnecting and its parallel solution algorithm. Int J Numer Methods Eng 32:1205–1227
17. Hajjar JF, Abel JF (1989) On the accuracy of some domain-by-domain algorithms for parallel processing of transient structural dynamics. Int J Numer Methods Eng 28:1855–1874
18. Hsieh S-H, Paulino GH, Abel JF (1995) Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis. Comput Methods Appl Mech Eng 121:137–162
19. Hsieh S-H, Paulino GH, Abel JF (1997) Evaluation of automatic domain partitioning algorithms for parallel finite element analysis. Int J Numer Methods Eng 40:1025–1051
20. Kim D-J, Duarte CA, Pereira JP (2008) Analysis of interacting cracks using the generalized finite element method with global-local enrichment functions. J Appl Mech 75(5):051107
21. Kim D-J, Pereira JP, Duarte CA (2010) Analysis of three-dimensional fracture mechanics problems: a two-scale approach using coarse generalized FEM meshes. Int J Numer Methods Eng 81(3):335–365. doi:10.1002/nme.2690
22. Kruis J, Matous K, Dostal Z (2002) Solving laminated plates by domain decomposition. Adv Eng Softw 33:445–452
23. Li S, Mear ME, Xiao L (1998) Symmetric weak-form integral equation method for three-dimensional fracture analysis. Comput Methods Appl Mech Eng 151:435–459
24. Melenk JM, Babuška I (1996) The partition of unity finite element method: Basic theory and applications. Comput Methods Appl Mech Eng 139:289–314
25. Moës N, Dolbow J, Belytschko T (1999) A finite element method for crack growth without remeshing. Int J Numer Methods Eng 46:131–150
26. Oden JT, Patra A, Feng YS (1992) An *hp* adaptive strategy. In: Noor AK (ed) Adapt Multilevel Hierarchical Comput Strateg. ASME, pp 23–46 (AMD-Vol. 157)
27. Oden JT, Duarte CA, Zienkiewicz OC (1998) A new cloud-based *hp* finite element method. Comput Methods Appl Mech Eng 153:117–126
28. O'Hara P, Duarte CA, Eason T (2009) Generalized finite element analysis of three-dimensional heat transfer problems exhibiting sharp thermal gradients. Comput Methods Appl Mech Eng 198(21–26):1857–1871. doi:10.1016/j.cma.2008.12.024
29. O'Hara P, Duarte CA, Eason T (2010) Transient analysis of sharp thermal gradients using coarse finite element meshes. Comput Methods Appl Mech Eng (submitted for publication)
30. OpenMP. Open multi-processing application programming interface. http://openmp.org/wp/
31. Pereira JP, Duarte CA, Guoy D, Jiao X (2009) *Hp*-Generalized FEM and crack surface representation for non-planar 3-D cracks. Int J Numer Methods Eng 77(5):601–633. doi:10.1002/nme.2419
32. Pereira JP, Duarte CA, Jiao X, Guoy D (2009) Generalized finite element method enrichment functions for curved singularities in 3D fracture mechanics problems. Comput Mech 44(1):73–92. doi:10.1007/s00466-008-0356-1
33. Quinn MJ (2004) Parallel programming in C with MPI and OpenMP. McGraw-Hill, New York
34. Raju JC, Newman IS Jr (1977) Three dimensional finite-element analysis of finite-thickness fracture specimens. Report TN D-8414, NASA-Langley Research Center, Hampton, VA, pp 1–40
35. Strouboulis T, Copps K, Babuška I (2001) The generalized finite element method. Comput Methods Appl Mech Eng 190:4081–4193
36. Sukumar N, Moës N, Moran B, Belytschko T (2000) Extended finite element method for three-dimensional crack modelling. Int J Numer Methods Eng 48(11):1549–1570
37. Top 500 supercomputer sites. November 2007 list. http://www.top500.org/list/2007/11/100
38. Ural A, Heber G, Wawrzynek P, Ingraffea A, Lewicki D, Neto J (2005) Three-dimensional, parallel, finite element simulation of fatigue crack growth in a spiral bevel pinion gear. Eng Fract Mech 72:1148–1170