Structural and Multidisciplinary Optimization: Vol. 47, pp. 49-61, 2013

Efficient Generation of Large-Scale Pareto-Optimal Topologies

Krishnan Suresh
University of Wisconsin, Madison
suresh@engr.wisc.edu

Abstract

The objective of this paper is to introduce an efficient algorithm and implementation for large-scale 3-D topology optimization. The proposed algorithm is an extension of a recently proposed 2-D *topological-sensitivity* based method that can generate numerous pareto-optimal topologies up to a desired volume fraction, in a single pass.

In this paper, we show how the computational challenges in 3-D can be overcome. In particular, we consider an arbitrary 3-D domain-space that is discretized via hexahedral/brick finite elements. Exploiting congruence between elements, we propose a matrix-free implementation of the finite element method. The latter exploits modern multi-core architectures to efficiently solve topology optimization problems involving millions of degrees of freedom. The proposed methodology is illustrated through numerical experiments; comparisons are made against previously published results.

1. INTRODUCTION

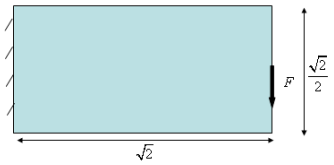
Consider the topology optimization problem:

$$\begin{aligned} &\underset{\Omega \subset D}{Min}\, J \\ &|\Omega| = v_0 |D| \end{aligned} \tag{1.1}$$

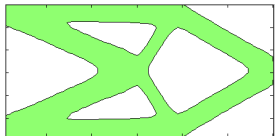
where:

$$\begin{aligned} &J: \text{ Compliance} \\ &\Omega: \text{ Geometry/topology to be computed} \\ &v_0: \text{ Desired volume fraction} \\ &D: \text{ Region within which the geometry must lie} \end{aligned} \tag{1.2}$$

Various methods such as homogenization [1], Solid Isotropic Material with Penalization (SIMP) [2] and level-set [3–6], have been proposed to solve such problems. These methods can systematically generate insightful topologies for complex engineering problems; see [7] and [8] for a review. For example, Figure 1a illustrates a plane-stress problem over a domain space D, with unit force; the material properties are: $E = 1$ and $\nu = 0.3$. An optimal topology Ω, generated via SIMP, for a volume fraction of 0.5 is illustrated in Figure 1b.



(a) Topology optimization problem



(b) Optimal topology for a volume fraction of 0.5

Figure 1: A typical topology optimization problem.

While the theory of topology optimization has reached a high level of maturity, one of the practical challenges that remain today is large-scale 3-D optimization. Such large-scale problems can take hours, or even days to complete.

The objective of this paper is to introduce a topology optimization method, and an implementation that dramatically reduces the computational time. The proposed method differs from SIMP in that it relies entirely on the concept of topological sensitivity. In Section 2, we review previous work on large-scale topology optimization. In Section 3, we discuss the theoretical and implementation aspects of the proposed strategy. In Section 4, numerical results are presented, followed by conclusions and open issues in Section 5.

2. LITERATURE REVIEW

In this Section, we review, in a chronological order, the strategies that have been proposed thus far to address large-scale 3-D topology optimization.

In one of the earliest work in this field [9], the authors relied on domain decomposition to parallelize the underlying finite element analysis in SIMP. A conjugate gradient iterative solver with Jacobi pre-conditioner was used to solve the linear system of equations. All implementations were carried out on a Cray T3E super computer. The authors demonstrated that the sensitivity calculations and the optimization process can also be parallelized. Topology optimization of 3-D problems with 1~3 million degrees of freedom, were completed in 3~40 hours (depending on the specific problem).

An alternate strategy based on design-space optimization was proposed by [10], where the finite element space is expanded as needed during the optimization process. This, the authors demonstrate, reduces the overall computational work-load, especially during the initial stages. As an industrial example, a 'knuckle-joint' was optimized, where the design starts with about 8000 elements, and is then expanded to 134,000 elements, taking a total of 115 hours.

In [11], the authors noted that in SIMP based topology optimization, as the density drops to zero, the linear system becomes ill-conditioned, and iterative solvers such as conjugate gradient or MINRES perform poorly. The authors therefore propose diagonal rescaling in combination with incomplete Cholesky preconditioning to address the ill-conditioning. Further, the vectors that span the Krylov (finite element) space are recycled during the iteration process. With these acceleration techniques, the authors demonstrated that, using an AMD Opteron TM252 2.6GHz 64-bit processor, with 8GB RAM, topology optimization with about 1 million degrees of freedom can be carried out in about 45 hours.

The importance of exploiting parallel-computing for topology optimization was recognized and explored in [12], where the authors used finite element tearing and interconnect (FETI) method with primal-dual solver. The authors once again identify the challenges posed by low

density values in SIMP on the rate of convergence, and provide workarounds.

In [13], the authors exploit approximate reanalysis, based on the combined approximation (CA) method [14] as an acceleration technique. Thus, with little or no loss in accuracy, a speed-up of 3~5 was achieved in 3-D.

In a more recent work [15], the authors implement topology optimization, in particular, the conjugate-gradient solver, on a graphics programmable unit (GPU). To exploit the hardware layout of the GPU, the authors rely on an assembly-free implementation of conjugate gradient. Since GPUs, at the time of publication of [15] were restricted largely to single-precision calculations, the conjugate gradient algorithm was modified to prevent accumulation of numerical errors. Furthermore, the implementation was restricted to a uniform-grid domain space (i.e., box-like design space) to exploit the architecture of GPUs. Given these restrictions, the authors achieved an impressive speed-up between 10~60 on a GeForce GTX280 with 1GB of memory compared to an optimized CPU code of the same algorithm. Using a GeForce 9800M GT, an older GPU card, a topology optimization problem over a 100x100x100 grid (i.e., about 3 million degree freedom) was solved in about 2 hours. The issue of slow convergence in SIMP due to low density values was not explicitly addressed.

A multi-resolution strategy was proposed in [16], where the authors adopt different discretization for finite element, density and design variables for SIMP-based optimization. This represents a shift in paradigm from the usual SIMP implementation. The authors show that one can arrive at the same topology for complex 3-D problems with significantly reduced computation, offering a new avenue of research for SIMP-based topology optimization. Specific timings for large-scale topology optimization were not provided.

Finally, a nested approach was proposed in [17] where by choosing convergence criteria (for the iterative solver) that are better tuned to the optimization objective (and sensitivities), it was shown that one can reduce the computational cost by about 60% without losing on accuracy. This was followed by the work reported in [18], where a single matrix factorization was used for the entire design process, in conjunction with iterative corrections. Consequently, the computational cost was reduced by one order of magnitude without any loss in accuracy.

While the focus of this paper is on large-scale topology optimization, we mention here recent advances that can address the deficiencies of SIMP. These include: (1) use of polygonal finite elements [19], [20] that circumvent anomalies such as checkerboard patterns, and are more adept at capturing optimal geometries, (2) an iso-geometric approach [21] where the material density is modeled via Non-Uniform Rational B-Spline basis functions, (3) mesh-independent projection techniques [22] to control internal and external feature sizes in topology optimization, and (4) combining continuum elements in SIMP with classic beam, plate and shell structural elements [23].

3. THE PARETO METHOD

3.1 Overview

In this paper, we build upon the 2-D pareto-topology-optimization (ParetO) method discussed in [24], [25] rather than SIMP, for three primary reasons:

1. ParetO does not rely on densities, i.e., all elements are either 'in' or 'out', and further, all the 'in' elements remain connected at all times, i.e., there is no 'hanging component' during the optimization process. Consequently, the stiffness matrices are inherently better conditioned. This leads to faster convergence of iterative solvers, i.e., the cost per finite element analysis is reduced, as illustrated later under Section 4.
2. For a desired volume fraction, it was observed in [24] that 2-D ParetO required fewer finite element operations than SIMP. For example, for the problem posed in Figure 1, the ParetO method required about half the number of finite element operation as the classic SIMP implementation [2].
3. Most importantly, ParetO, as the name suggests, is designed to solve a multi-objective problem:

$$\underset{\Omega \subset D}{Min}\{J, |\Omega|\} \tag{3.1}$$

In other words, it can find numerous pareto-optimal topologies up to a desired volume fraction *with no additional cost*; see Figure 2. Such pareto-optimal topologies can be particularly beneficial to the design engineer for large-scale problems where one cannot afford to repeatedly solve topology optimization problems, at various volume fractions.

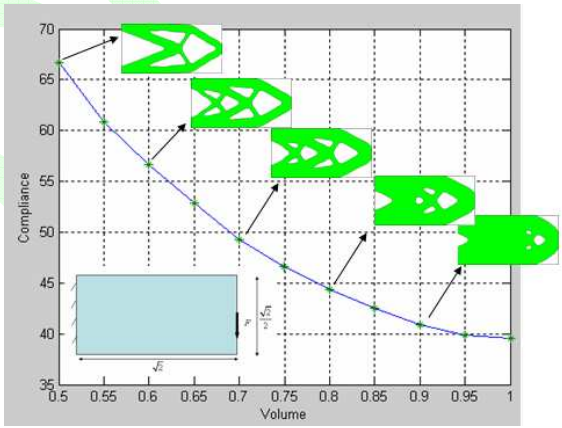


Figure 2: The pareto-optimal curve and topologies.

Given these inherent advantages, we consider here extending the 2-D ParetO method and algorithm to large-scale 3-D problems. The specific challenges that one faces in 3-D, and solution strategies are discussed in the remainder of this paper.

3.2 Finite Element Mesh

The initial design space can be of arbitrary size and shape, and is discretized into tri-linear hexahedral elements; the latter offer a good compromise between accuracy and speed. Two illustrative examples are shown in Figure 3; the example on the left is that of a uniform structured grid, while the other is that of a non-uniform grid over an arbitrary design space. The ability to handle arbitrary design spaces is critical for large scale industrial applications. The finite-element shape-functions and element-stiffness matrices for tri-linear hexahedral elements can be found, for example, in [26].

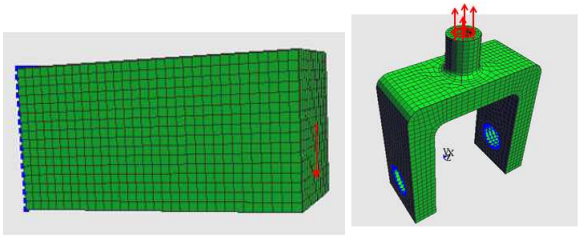


Figure 3: Examples of domain space discretized via tri-linear hexahedral/brick elements.

Typically, in finite element analysis (FEA), the global stiffness matrix K is assembled prior to a linear-solve. However, in the present work, we have chosen an 'assembly-free' (a.k.a. 'matrix-free') approach [27] where the global stiffness matrix K is not assembled or stored. Instead, only the unique individual element stiffness matrices are computed and stored in memory. Not only does this reduce the overall memory requirement (this is especially important for GPU), it can also accelerate topology optimization since FEA is largely memory-bandwidth limited [28].

Computing the unique elements is trivial if the finite element mesh is a uniform grid mesh (see Figure 3a). However, consider the non-uniform mesh such as the one in Figure 3b. The specific strategy adopted in this paper to reduce memory consumption is to exploit congruency of mesh-elements. In other words, we only compute and store the stiffness matrix of geometrically and materially distinct mesh-elements. In the next Section, we describe an algorithm for detecting congruency between mesh elements.

3.3 Mesh-Element Congruency

The central question is the following: Given two hexahedral elements (such as the ones in Figure 4), will they result in the same element matrix?

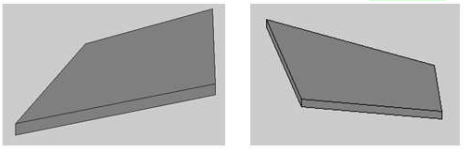


Figure 4: Two possibly congruent mesh elements.

Observe that the element stiffness matrix is rigid-body invariant. Thus, assuming, for simplicity, that the material properties are the same for both elements, then the two elements will yield the same the element stiffness matrix if they are geometrically congruent. Towards this end, we recall that the following theorem [29].

Theorem (Cauchy, 1839): *Two convex polyhedra with corresponding congruent and similarly situated faces have equal corresponding dihedral angles.*

The theorem essentially states that 12 edge-length measurements are sufficient to establish congruency. Towards this end, a hashing function [30] is created that takes 12 doubles, and returns a long integer; the latter is referred to as the element-signature Various hashing functions can be constructed [30]; in this paper, a weighted average of the edge-lengths is used. Consequently, two elements are congruent if their element-signatures are identical (to desired precision). Thus, we first compute the element-signatures and perform a sorting operation to identify congruency. Only the element stiffness matrices of distinct elements are computed and stored. This step needs to be performed once at the beginning of topology optimization.

3.4 Matrix-Free Krylov Iterations

A matrix-free implementation of Krylov-space algorithms for solving a linear system of equations is straight-forward [27]. Specifically, all Krylov-space algorithms require a matrix-vector multiplication Ku where K is the global stiffness matrix and u is the global solution vector. In a matrix-free implementation, we have:

$$Ku = \left(\sum_e K_e \right) u = \sum_e K_e u_e \tag{3.2}$$

A particular Krylov-space algorithm namely, the Jacobi-preconditioned conjugate-gradient (Jacobi-PCG) method [31] is employed here. Alternate pre-conditioners such as incomplete-LU can reduce the total number of iterations, but their overall benefit in a highly parallel environment is questionable since the cost of incomplete-LU factorization can dominate in large-scale problems. Furthermore, in the PareTO method, the stiffness matrix is well-conditioned, and consequently, even the simple Jacobi-PCG converges rapidly, and is well suited for multi-core architectures. Details of convergence are provided in Section 4.

3.5 Topological Sensitivity Computation

Once FEA is carried out in an assembly-free manner, the *topological sensitivity* in each mesh-element is computed as described below.

Topological sensitivity captures the first order impact of inserting a small circular hole within a domain on various quantities of interest. This concept has its roots in the influential paper by Eschenauer, et. al. [32], and has later been extended and explored by numerous authors [33–37]. Generalization of topological sensitivity to arbitrary features has been addressed, for example, in [38–40].

For compliance J of a 2-D plane-stress problem, a closed-form expression for the topological sensitivity is given by [41]:

$$\mathcal{T}(p) = \frac{4}{1+\nu}\sigma : \varepsilon - \frac{1-3\nu}{1-\nu^2} tr(\sigma) tr(\varepsilon) \tag{3.3}$$

Thus, given the stress and strain field in a domain, one can compute the topological sensitivity *field* (that is location dependent). Figure 5 illustrates the topological sensitivity (TS) field for the full-domain D for the problem in Figure 1a. Observe that, at the bottom-right, the TS-field is 'o', implying that removing material at that point has little impact on the compliance. On the other hand, the TS-field at bottom left is approximately 1.0 unit, implying that inserting a hole at that point will increase the compliance by 1 unit (multiplied by the size of the hole).

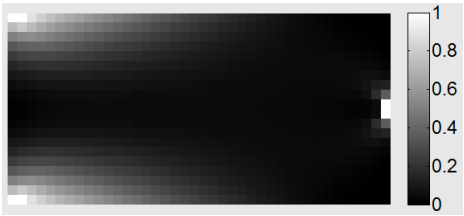


Figure 5: A (scaled) compliance topological sensitivity field for the problem in Figure 1a.

The generality of the proposed method is inherited from the generality of the topological sensitivity field; the latter is well-defined, and can be computed for various quantities of interest In 3-D, the topological sensitivity field for compliance is given by [34]:

$$\mathcal{T} = -20\mu\sigma : \varepsilon - (3\lambda - 2\mu)tr(\sigma)tr(\varepsilon) \qquad (3.4)$$

where μ & λ are the Lame parameters.

For eigen-value problems, one can show that [42], [43]:

$$\mathcal{T} = \sigma : \varepsilon - \omega_n^2 \rho \|u_n\|^2 \qquad (3.5)$$

where ρ is the material density, σ & ε are the stress & strain tensors, and ω_n is the corresponding eigen-value. Figure 6 illustrates the TS field corresponding to the 1st eigen-mode of the domain of Figure 1a, with $\rho = 1$. The similarities between Figure 6 and Figure 5 are worth noting.

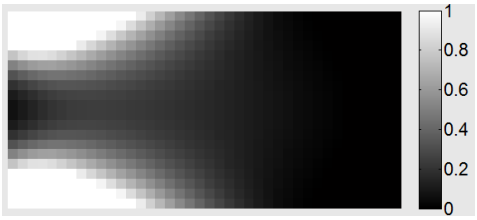


Figure 6: A (scaled) topological sensitivity field for the 1st eigen-mode for the domain in Figure 1a.

3.6 Topological Sensitivity Field as a Level-Set

A naïve approach to exploiting topological sensitivity (TS) field is to 'kill' mesh-elements with low TS values. However, this leads to instability and checker-board patterns. Alternately, one can exploit the TS field, in conjunction with other level-set fields to introduce holes during the topology optimization process [44]. Finally, the authors in [45], use the TS field with conjunction with fictitious domain method to enhance the convergence of topology optimization.

Here we propose an alternate and a more powerful approach where one directly exploits the TS field as a level-set. To illustrate, Figure 7 shows the compliance TS field of Figure 5 as 3-dimensional surfaces. Also illustrated in Figure 7 are two cutting planes corresponding to $\tau = 0.01$ and $\tau = 0.03$, respectively.

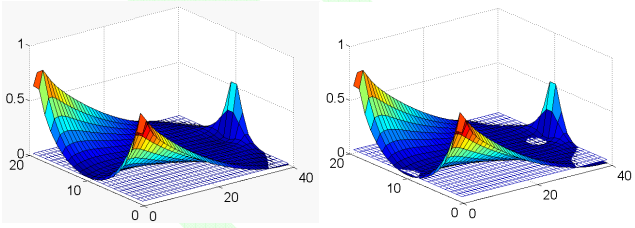


Figure 7: TS-fields and cutting-planes $\tau = 0.01$ and $\tau = 0.03$.

As with any level-set, the cutting-planes induce a domain Ω^τ defined per:

$$\Omega^\tau = \{p \mid \mathcal{T}(p) > \tau\} \qquad (3.6)$$

In other words, the domain Ω^τ is the set of all points where the TS field exceeds the prescribed value of τ. The induced domains Ω^τ, corresponding to $\tau = 0.01$ and $\tau = 0.03$ are illustrated in Figure 8. Observe how portions of the domain with low TS values have been eliminated.

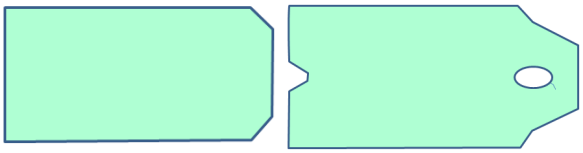


Figure 8: The induced domains Ω^τ corresponding to $\tau = 0.01$ and $\tau = 0.03$.

If $\tau = 0$ (the lowest value of the TS field in this instance), then Ω^τ will coincide with Ω. The notion of a 'cutting-plane' generalizes to a cutting-manifold in 3-D, and Equation (3.6) applies to 3-D as well. The PareTO algorithm discussed in the next Section relies on computing the value of τ for a desired volume fraction through a fixed-point iteration (proposed in [37], and also adopted in [45]).

3.7 PareTO Algorithm

We now have the necessary ingredient to address the overall PareTO algorithm illustrated in Figure 9; each of the steps is described below.

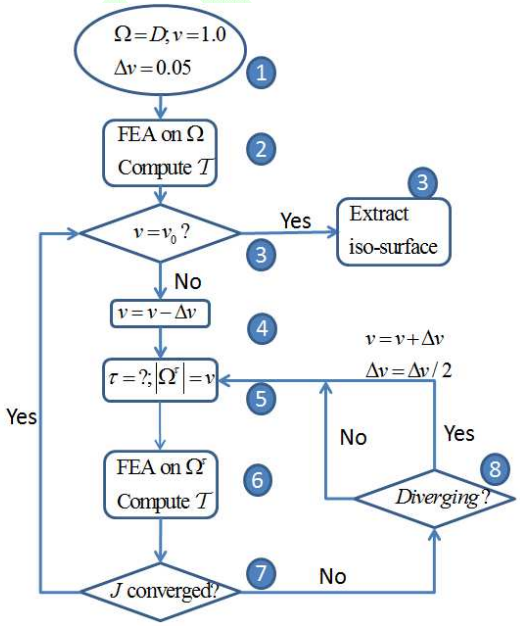


Figure 9: Proposed 3-D topology optimization algorithm.

1. We start with $\Omega = D$, i.e., we start with a volume fraction of 1.0. Observe that this is different from SIMP, where a constant density ρ is assigned to the design-space D such that the 'apparent' volume fraction throughout the optimization process is v_0. The cutting-plane parameter τ is initialized to zero.
2. Next, we carry out a finite element analysis on D and compute the TS field as discussed earlier.
3. If the desired volume fraction has been reached, the iso-surface with the current cutting-plane value τ is extracted. For iso-surface extraction, we rely on the classic marching-cubes algorithm described in [46], where the TS values at the corner nodes of the hex mesh are used to extract the iso-surface.
4. (Else) We decrement the current volume fraction by Δv; Δv is initialized to 0.05, but this is controlled in an adaptive fashion (see step 8 below).

5. Given the current TS field and the target volume fraction, we seek the parameter τ such that $|\Omega^\tau|$ is equal to the target volume fraction. This is a simple binary-search algorithm where maximum and minimum values of the TS-field serve as the limits of the binary search.
6. Once the desired value of τ has been computed, a finite element analysis is carried out on Ω^τ (where elements that lie outside are not included in the FEA) and, the TS field is recomputed.
7. If the τ value has converged (to within user defined accuracy) we return to Step-3. If the parameter has not yet converged, we return to Step 5, after performing the check below to ensure that the optimization process is not diverging.
8. If a very large step size Δv is specified by the user, the above process may diverge. If this is detected (by diverging values of compliance), the value of Δv is reduced by a factor of 2, prior to returning to Step 5.

4. NUMERICAL EXPERIMENTS

In this Section, we present results from numerical experiments based on the above algorithm. The default parameters are as follows:

- As described earlier, the domain D can be of arbitrary shape and size, and is discretized via tri-linear hex/brick elements; the elements conform to the boundary of D (unless otherwise noted).
- The material properties are $E = 1$ and $\nu = 0.3$
- The residual for the preconditioned conjugate gradient is set to 1e-8, unless otherwise noted.
- The volume step-size (Step-1 of see Figure 9) was set to 0.05, unless otherwise noted.
- In Step-7 of the algorithm (see Figure 9), the fixed-point iteration is assumed to have converged if the change in compliance is less than 1%.

All experiments were conducted on a Windows 7 64-bit machine with the following hardware:

- Intel I7 960 CPU quad-core running at 3.2GHz with 6 GB of memory; parallelization of CPU code was implemented through OpenMP commands.
- The graphics programmable unit (GPU) is an Nvidia GeForce GTX 480 (480 cores) with 1.5 GB.
- Both the CPU and GPU were configured to run in double-precision.

4.1 Point-Load Cantilever: CG Iterations

The first experiment involves the classic cantilever beam illustrated in Figure 10, where one end of the beam is fixed, while a tip load is applied at the other end. A typical hex-mesh consisting of 32x16x16 elements is shown in Figure 10, i.e., with 28611 degrees of freedom (DOF).

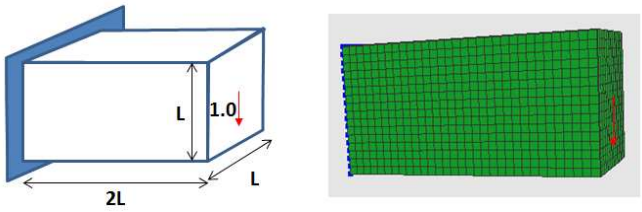


Figure 10: The cantilevered beam problem and hex-mesh.

The optimal topology for a volume fraction of 0.5 was computed with a step-size of 0.05 using the CPU and GPU (two independent runs); Figure 11 illustrates the optimal topology, with a compliance of 4.88 units; identical results were obtained from CPU and GPU.

On the CPU, the optimal topology was computed in 9.04 seconds. On the GPU, the computational time was reduced to 6.04 seconds, i.e., overall speed-up was approximately 1.5. Both implementations are otherwise identical, requiring 30 finite element iterations, in total.

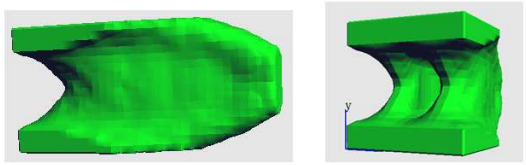


Figure 11: Optimal topology for a volume fraction of 0.5 (side and rear views).

In the PareTO method, recall that an element is either 'in' or 'out', i.e., there is no concept of density. However, we artificially forced the relative-density of 'out' elements to 0.005, i.e., $E = 1.25e^{-7}E_0$ and studied the convergence.

Figure 12 illustrates the impact of this change on the CG iterations across the 30 finite element runs. A pure 'in/out' approach requires an almost constant number of iterations, i.e., CG is relatively insensitive to the absence of material. On the other hand, with a non-zero density, the number of CG iterations increases as elements are removed from the design space. This can be observed in Figure 12. One can also observe the oscillation in the number of iterations. This is consistent with the algorithm in Figure 9 since the solution is recycled in Steps 5 through Step 8, and therefore fewer CG iterations are needed. This experiment highlights one the primary advantages of the PareTO method.

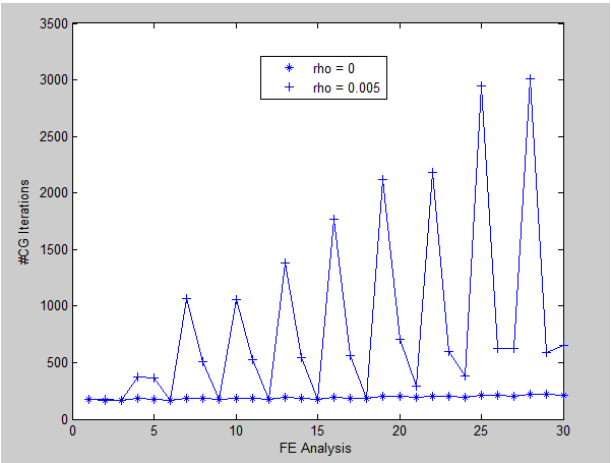


Figure 12: Impact of density on CG iterations.

4.2 Point-Load Cantilever: Verification

The dimensions of the above cantilevered beam problem were modified as shown in Figure 13; this corresponds to the problem considered in [9]. The mesh now consists of 128x80x24 elements, i.e., 783,675 DOF.

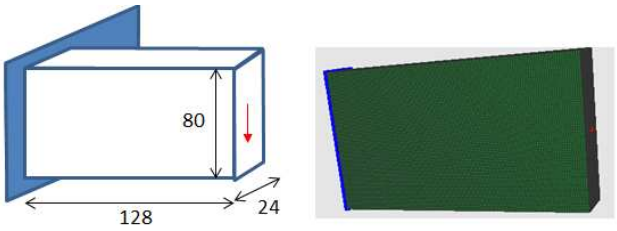


Figure 13: The cantilevered beam problem and hex-mesh with 783K DOF.

Figure 14 illustrates the computed and published topologies [9] for a volume fraction of 0.5. While the overall shape is similar, the detailed topologies are different. This is not surprising since: (1) it is well known that topology optimization does not yield unique solutions, and (2) the algorithms are fundamentally different, and (3) so are the convergence criteria, iso-surface extraction, etc.

The time reported in [9] for solving this problem, by exploiting symmetry, is 3.9 hours. In the present work, with a volume step-size of 0.05, the computational times are as follows: 16 minutes (CPU) and 125 seconds (GPU); we did not exploit symmetry.

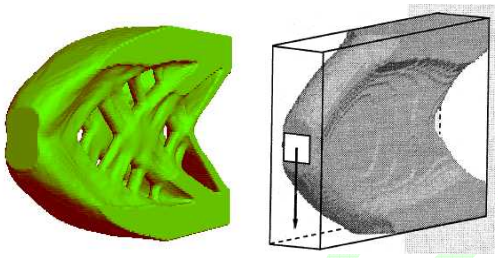


Figure 14: (a) Computed topology, (b) computed by [9].

The initial compliance computed in the present work is 1 unit, and the final compliance is 1.47 units; the compliance was not reported in [9]. The compliance versus volume fraction for the present work is illustrated in Figure 15. Note that typically 3 finite element iterations are required to move from one volume step to the next. For this problem, a total of 29 finite elements operations are required.

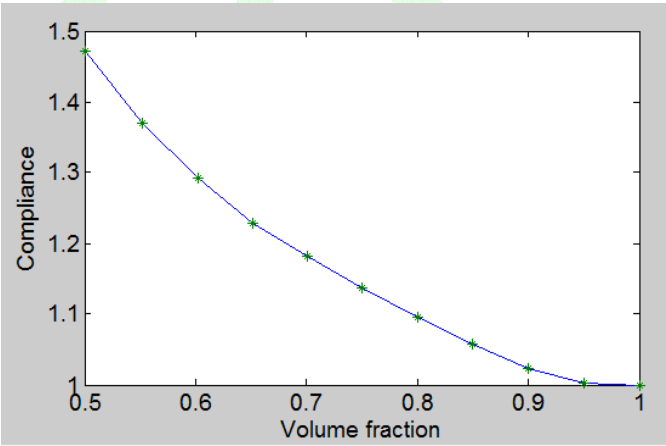


Figure 15: Compliance versus volume fraction pareto-curve.

4.3 Edge-Load Cantilever: Verification

The next experiment involves the domain illustrated in Figure 16, consisting of 84x28x14 elements, i.e., 110925 DOF; this problem was considered in [11].

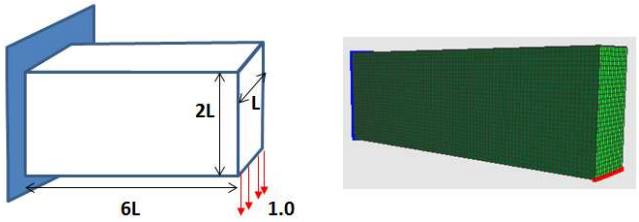


Figure 16: Edge-loaded cantilevered beam and hex-mesh with 110925 DOF.

Figure 17 illustrates the topology computed here for a volume fraction of 0.5. The final topology was computed after 26 finite element operations, in 200 seconds on the CPU, and 44.8 seconds, on the GPU. The ratio of final to initial compliance is 1.37. The computational time reported in [11] is 2.4 hours.

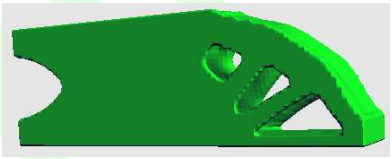
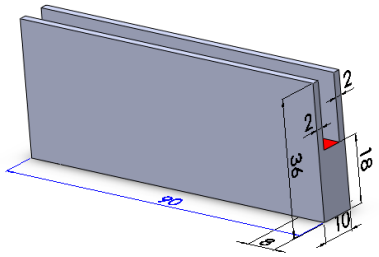


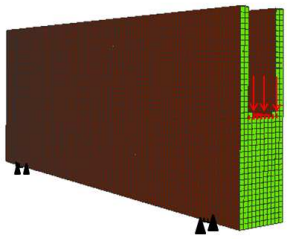
Figure 17: Optimal topology for the problem in Figure 10, obtained in 200 seconds (CPU) and 44.8 seconds (GPU).

4.4 Bridge Design: Pareto Topologies

We now consider the bridge problem illustrated in Figure 18, where a uniform load is applied on a horizontal layer, and the design space is supported at four symmetric points at a distance of 8 units from the two ends. A hex-mesh with approximately 113,000 dof is also shown in Figure 18.



(a) Bridge dimensions.

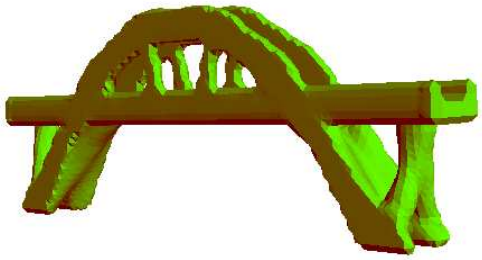


(b) Hex mesh with 113,000 dof.

Figure 18: The bridge problem.

Figure 19 illustrates the optimal bridge for a volume fraction of 0.35. The final topology was computed after 32 finite element operations, in 120 seconds on the CPU, and 36.2 seconds, on the GPU. For visual comparison, Figure 17 also illustrates the Oregon City Bridge spanning the Willamette

River; the similarities are striking. Such observations have been made by other authors as well [16].



(a) Optimal bridge for volume fraction of 0.35

(b) Oregon city bridge.

Figure 19: Simulated and real bridge designs.

Recall that the PareTO method computes all optimal topologies up to a desired volume fraction, at no additional cost. Figure 20 illustrates the computed bridge for a volume fraction of 0.40.

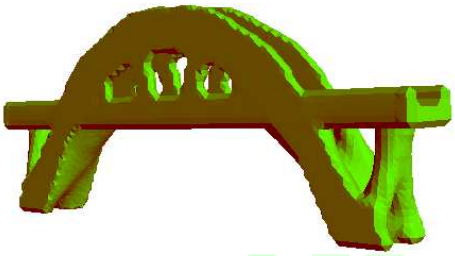


Figure 20: Optimal bridge for a volume fraction of 0.40

4.5 Knuckle Design: Non-uniform Hex-mesh

To illustrate the generality of the implementation, we consider optimizing the topology of a 'knuckle' illustrated in Figure 21a. Unlike the previous examples, the hex-mesh is irregular and consists of 20160 degrees of freedom; the mesh was created using Abaqus, a commercial FEA system. Figure 21b illustrates the optimal topology for a volume fraction of 0.55. The total computational times are 109 seconds (CPU) and 42 seconds (GPU).

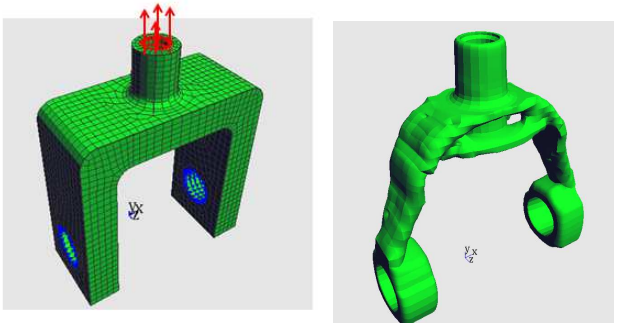


Figure 21: (a) A knuckle problem and (b) optimal topology

4.6 Stool Design: Insensitivity to Mesh-Size

Next we study the impact of discretization on the final topology. As a specific example, we consider the 'stool design' problem illustrated in Figure 22 where the vertical displacement of the four corners is restrained, and a vertical force is applied at the center as shown. In addition, to eliminate singularity, the horizontal displacements of the one of the corners are also restrained. This problem was also solved by the authors in [9].

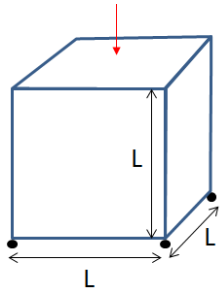
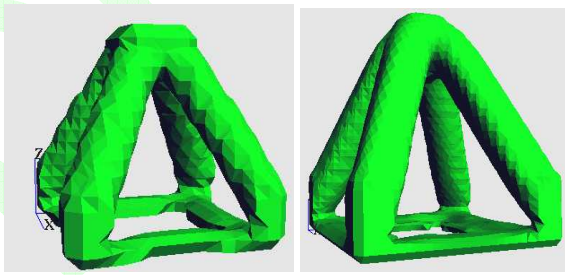
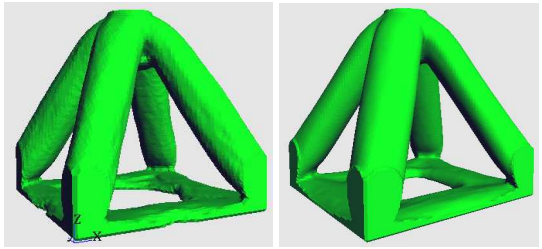


Figure 22: Stool design problem.

The domain was discretized into NxNxN elements where N was varied. Figure 23 illustrates the optimal designs for a volume fraction of 0.2, with a step-size of 0.10, for four different values of N. The GPU computing time, and final-to-initial compliance ratio for each case is also listed.



(a) N=16; t=5.8s;J/J_0=1.85 (b) N=32; t=19s; J/J_0=1.43



(c) N=64; t=185s; J/J_0=1.39 (d) N=96; t=850s; J/J_0=1.39

Figure 23: Optimal stool-designs

Besides a refinement in topology and detail, the algorithm leads to similar designs. The speed-up offered by the GPU over the CPU varied from 1.5 for N =16, to 6, for N = 96.

4.7 Table Design: Insensitivity to Step-Size

In the PareTO algorithm of Figure 9 we start with an initial step-size of 0.05. Here, we highlight the adaptive nature of the algorithm if a different step-size is prescribed. In other words, suppose the user specifies a step-size of 0.25 in order to accelerate computation, the algorithm may fail to converge in the first attempt. This is automatically detected, and the algorithm makes a second attempt with a step-size of 0.125 and so on.

As a specific example, consider the 'table design' problem illustrated in Figure 24 where the four corners are restrained, and a vertical pressure is applied on the top-face.

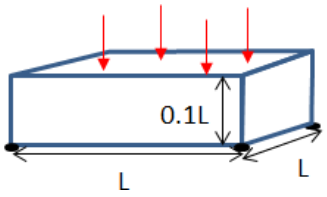
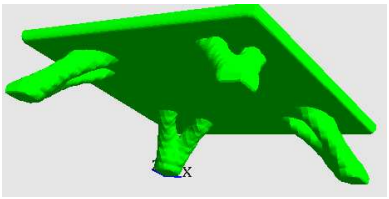
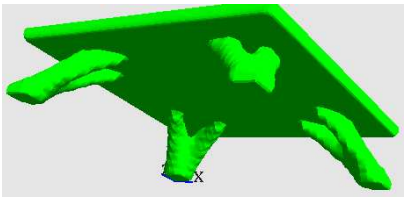


Figure 24: Table design problem.

The domain is discretized into 8*80*80 elements. Figure 25 illustrates the optimal topologies for a volume fraction of 0.2, with two different initial step-sizes. Observe that, the computational time is larger with a larger initial step-size since numerous FEA are 'wasted' initially. However, in both cases, the final topologies are almost identical with a compliance difference of less than 1%.



(a) $\Delta v_0 = 0.50$; t=93 secs; J = 5.82.



(b) $\Delta v_0 = 0.05$; t=71 secs; J = 5.76.

Figure 25: Optimal table-designs for volume fraction of 0.2

Since the computational cost can prohibit extremely small steps, we recommend an initial step size of 0.05. If the method fails to converge for this step-size, a smaller step will be taken in an adaptive fashion. Further theoretical investigation is required to understand the impact of step-size on the final topology and objective function.

4.8 Case Study: Airline Passenger Seat Frame

Next consider the design of an airline passenger seat frame illustrated schematically in Figure 26 [47]. While air-frames must meet numerous objectives: low mass, high stiffness, crash-worthiness, low-cost, etc., we focus here on weight and compliance minimization.

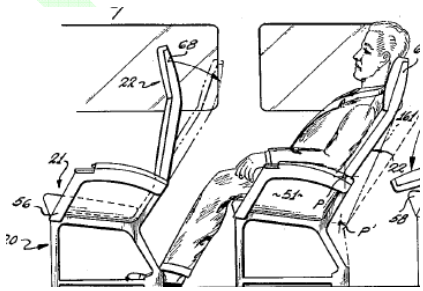


Figure 26: A schematic of a passenger airline frame.

The initial geometry is illustrated in Figure 27a (all linear units in mm); the material is chosen to be 1060 aluminum ($E = 69GPa; \upsilon = 0.33$). The applied boundary conditions are illustrated in Figure 27b where it is assumed that the load on the horizontal plane to back-support is in the ratio 4:1. The front edge is fixed, while the back edge is fixed in the vertical direction.

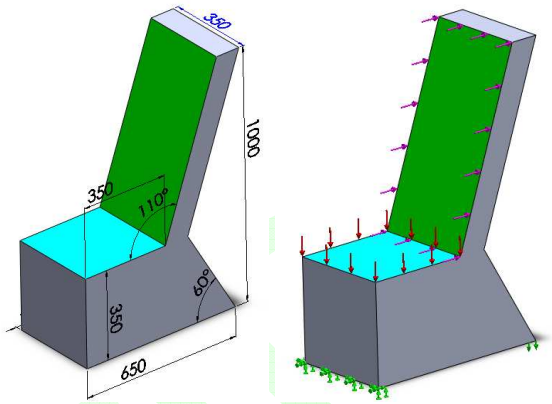


Figure 27: (a) The initial design space and (b) Boundary conditions for a seat-frame.

The geometry was discretized using a *non-conforming* uniform grid, resulting in 56,000 elements and 63,000 nodes, i.e., 189,000 degrees of freedom.

Designs for various volume fractions were obtained; two of which are illustrated in Figure 28 for volume fraction of 0.3 and 0.20. Observe that the 0.30 design is closed at the back, and does not provide access to the space underneath and therefore undesirable. Other manufacturing and ergonomic constraints must be included to improve the final design.

Figure 28: Optimal seat frames for fractions of 0.3 and 0.2.

4.9 Summary of Computational Times

In Table 1, we summarize results for some of the large-scale examples, and compare them against published data, wherever available. The PareTO columns include: (i) the total number of FEA iterations taken to reach the final volume fraction, (ii) the ratio of final to initial compliance, (iii) the CPU time and (iv) the GPU time.

Both the CPU and GPU implementations are restricted by available memory. In the GPU, with 1.5 GB of memory, the tip-cantilever problem, with 15 million degrees of freedom, was solved in 2 hours. In the CPU, with 6 GB of memory, the same problem, but with 92 million degrees of freedom was solved in approximately 12 days (as summarized below).

Table 1: Summary of computing times and compliances.

Name of part & volume fraction	**DOF**	**Pub. Data**	**PareTO**			
			#FEA	$\frac{J}{J_0}$	**CPU**	**GPU**
Cantilever Beam; Edge (0.50)	110K	2.4 hr, #FEA: 139 [11]	26	1.37	200 secs	45 secs
Knuckle (0.55)	20K	--	24	1.23	111 secs	44 secs
Bridge (0.35)	113K	--	32	1.45	2 mins	36.2 secs
Stool; N=96 (0.20)	2.7M	21.8 hr; #FEA: 703 [9];	23	1.43	1 hr, 24 mins	14 mins
Point Load Cantilever (0.50)	783K	3.9 hr, #FEA: 470 [9];	29	1.47	16 mins	125 s
	15M	-	22	1.53	19hr, 28 mins	2hr, 12 mins
	92M	-	23	1.52	12 days, 2hr	-

5. CONCLUSIONS

The main contribution of the paper is an efficient, and yet simple algorithm for large scale 3-D topology optimization. As illustrated via numerical examples, the proposed algorithm and implementation is 1~2 orders of magnitude faster than previously published literature. Readers may download PareTO from www.ersl.wisc.edu.

The work presented here has largely focused on compliance minimization. However, as demonstrated, the PareTO method is applicable for other problems as well [43]. It remains to be seen if this can be successfully extended to other classes of problems. Further, we have not attempted here to demonstrate either the parallel or numerical scalability of the method [12], and we have used a simple Jacobi preconditioned conjugate gradient; we hope to address these limitations in the future. Indeed, if the acceleration techniques reviewed earlier in the paper can be incorporated, the method can perhaps be further improved.

Nonlinear problems in topology optimization have received significant attention lately [48]. Extending PareTO to such nonlinear problems is a promising avenue since the topological sensitivity concept has been generalized to nonlinear problems as well [49].

6. REFERENCES

[1] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method," *Computer Methods in Applied Mechanics and Engineering*, vol. 71, pp. 197–224, 1988.

[2] O. Sigmund, "A 99 line topology optimization code written in Matlab," *Structural and Multidisciplinary Optimization*, vol. 21, no. 2, pp. 120–127, 2001.

[3] G. Allaire, "A level-set method for shape optimization," *Comptes Rendus Mathematique*, vol. 334, no. 12, pp. 1125–1130, 2002.

[4] G. Allaire and F. Jouve, "A level-set method for vibration and multiple loads structural optimization," *Structural and Design Optimization*, vol. 194, no. 30–33, pp. 3269–3290, 2005.

[5] L. He, "Incorporating topological derivatives into shape derivatives based level set methods," *Journal of Computational Physics*, vol. 225, no. 1, pp. 891–909, 2007.

[6] M. Y. Wang, "A level set method for structural topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, pp. 227–246, 2003.

[7] G. Rozvany, "A critical review of established methods of structural topology optimization," *Structural and Multidisciplinary Optimization*, vol. 37, no. 3, pp. 217–237, 2009.

[8] M. P. Bendsoe and O. Sigmund, *Topology Optimization: Theory, Methods and Application*, 2nd ed. Springer, 2003.

[9] T. Borrvall and J. Petersson, "Large-scale topology optimization in 3-D using parallel computing," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, pp. 6201–6229, 2001.

[10] Y. I. Kim and B. M. Kwak, "Design space optimization using a numerical design continuation method," *International Journal for Numerical Methods in Engineering*, vol. 53, no. 1979–2002, 2002.

[11] S. Wang, E. D. Sturler, and G. Paulino, "Large-scale topology optimization using preconditioned Krylov subspace methods with recycling," *International Journal for Numerical Methods in Engineering*, vol. 69, no. 12, pp. 2441–2468, 2007.

[12] A. Evgrafov, C. J. Rupp, K. Maute, and M. L. Dunn, "Large-scale parallel topology optimization using a dual-primal substructuring solver," *Structural and Multidisciplinary Optimization*, vol. 36, pp. 329–345, 2008.

[13] O. Amir, M. Bendsøe, and O. Sigmund, "Approximate reanalysis in topology optimization," *International Journal for Numerical Methods in Engineering*, vol. 78, pp. 1474–1491, 2009.

[14] U. Kirsch, "Combined Approximations - A General Reanalysis Approach for Structural Optimization," *Structural and Multidisciplinary Optimization*, vol. 20, no. 2, pp. 97–106, 2000.

[15] S. Schmidt, "A 2589 Line Topology Optimization Code Written for the Graphics Card," Univeritat Trier; www.am.uni-erlangen.de, Technical report Preprint SPP1253-068, 2009.

[16] T. Nguyen, G. Paulino, J. Song, and C. Le, "A computational paradigm for multiresolution topology optimization (MTOP)," in *Structural and Multidisciplinary Optimization*, vol. 41, 2010, pp. 525–539.

[17] O. Amir, M. Stolpe, and O. Sigmund, "Efficient use of iterative solvers in nested topology optimization," *Structural and Multidisciplinary Optimization*, vol. 42, no. 1, pp. 55–72, 2010.

[18] O. Amir and O. Sigmund, "On reducing computational effort in topology optimization: how

far can we go?," *Structural and Multidisciplinary Optimization*, vol. 44, no. 1, pp. 25–29, 2011.

[19] C. Talischi, G. Paulino, A. Pereira, and F. M. Menezes, "Polygonal finite elements for topology optimization: A unifying paradigm," *International Journal for Numerical Methods in Engineering*, vol. 82, no. 6, pp. 671–698, 2010.

[20] C. Talischi, G. Paulino, A. Pereira, and F. M. Menezes, "PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes.," *Structural and Multidisciplinary Optimization*, vol. Published Online. DOI: 10.1007/s00158–011–0696-x, 2012.

[21] B. Hassani, M. Khanzadi, and S. M. Tavakkoli, "An isogeometrical approach to structural topology optimization by optimality criteria," *Structural and Multidisciplinary Optimization*, vol. 45, no. 2, pp. 223–233, 2012.

[22] S. R. M. Almeida, G. Paulino, and E. C. N. Silva, "A Simple and Effective Inverse Projection Scheme for Void Distribution Control in Topology Optimization," *Structural and Multidisciplinary Optimization*, vol. 39, no. 4, pp. 359–371, 2009.

[23] L. L. Stromberg, A. Beghini, W. F. Baker, and G. Paulino, "Topology Optimization for Braced Frames: Combining Continuum and Discrete Elements," *Engineering Structures*, vol. 37, pp. 106–124, 2012.

[24] K. Suresh, "A 199-line Matlab code for Pareto-optimal tracing in topology optimization," *Structural and Multidisciplinary Optimization*, vol. 42, no. 5, pp. 665–679, 2010.

[25] I. Turevsky and K. Suresh, "Efficient Generation of Pareto-Optimal Topologies for Compliance**," *International Journal for Numerical Methods in Engineering*, vol. 87, no. 12, pp. 1207–1228, 2011.

[26] O. C. Zienkiewicz, *The Finite Element Method for Solid and Structural Mechanics*. Elsevier, 2005.

[27] C. E. Augarde, A. Ramage, and J. Staudacher, "An element-based displacement preconditioner for linear elasticity problems," *Computers and Structures*, vol. 84, no. 31–32, pp. 2306–2315, 2006.

[28] D. Goddeke, R. Strzodka, and S. Turek, "Performance and accuracy of hardware-oriented native-emulated- and mixed-precision solvers in FEM simulations," *International Journal of Parallel , Emergent and Distributed Systems*, vol. 22, no. 4, pp. 221–256, 2007.

[29] A. Borisov, M. Dickinson, and S. Hastings, "A Congruence Problem for Polyhedra," *The American Mathematical Monthly*, vol. 117, no. 3, pp. 232–249, 2010.

[30] W. H. Press and et. al., *Numerical Recipies: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[31] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[32] H. A. Eschenauer, "Bubble method for topology and shape optimization of structures," *Structural Optimization*, vol. 8, pp. 42–51, 1994.

[33] A. A. Novotny, "Topological Derivative for Linear Elastic Plate Bending Problems," *Control and Cybernetics*, vol. 34, no. 1, pp. 339–361, 2005.

[34] A. A. Novotny, "Topological Sensitivity Analysis for Three-dimensional Linear Elasticity Problem," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 41–44, pp. 4354–4364, 2005.

[35] A. A. Novotny, "Topological-Shape Sensitivity Method: Theory and Applications," *Solid Mechanics and its Applications*, vol. 137, pp. 469–478, 2006.

[36] J. Sokolowski, "On Topological Derivative in Shape Optimization," *SIAM journal on control and optimization*, vol. 37, no. 4, pp. 1251–1272, 1999.

[37] J. Céa, S. Garreau, P. Guillaume, and M. Masmoudi, "The shape and topological optimization connection," *Computer Methods in Applied Mechanics and Engineering*, vol. 188, no. 4, pp. 713–726, 2000.

[38] I. Turevsky, S. H. Gopalakrishnan, and K. Suresh, "An Efficient Numerical Method for Computing the Topological Sensitivity of Arbitrary Shaped Features in Plate Bending**," *International Journal of Numerical Methods in Engineering*, vol. 79, pp. 1683–1702, 2009.

[39] I. Turevsky and K. Suresh, "Generalization of Topological Sensitivity and its Application to Defeaturing**," in *ASME IDETC Conference*, Las Vegas, 2007.

[40] S. H. Gopalakrishnan and K. Suresh, "Feature Sensitivity: A Generalization of Topological Sensitivity**," *Finite Elements in Analysis and Design*, vol. 44, no. 11, pp. 696–704, 2008.

[41] R. A. Feijóo, "The topological-shape sensitivity method in two-dimensional linear elasticity topology design," in *Applications of Computational Mechanics in Structures and Fluids*, V. S. S.R. Idelsohn, Ed. CIMNE, 2005.

[42] G. Allaire, "A level-set method for vibration and multiple loads structural optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 3269–3290, 2005.

[43] I. Turevsky and K. Suresh, "Tracing the Envelope of the Objective-Space in Multi-Objective Topology Optimization**," presented at the ASME IDETC/CIE Conference, Washington, DC, 2011.

[44] G. Allaire, "Structural Optimization using Sensitivity Analysis and a Level-set Method," *Journal of Computational Physics*, vol. 194, no. 1, pp. 363–393, 2004.

[45] J. A. Norato, "A topological derivative method for topology optimization," *Structural and Multidisciplinary Optimization*, vol. 33, pp. 375–386, 2007.

[46] W. E. Lorensen and H. E. Cline, "Marching Cubes: a high resolution 3D surface reconstruction algorithm.," *Computer Graphics (Proc. of SIGGRAPH)*, vol. 21, no. 4, pp. 163–169, 1987.

[47] B. F. Monroe, "Aircraft seat structure," U.S. Patent 30378121962.

[48] G. H. Yoon, "Maximizing the fundamental eigenfrequency of geometrically nonlinear structures by topology optimization based on element connectivity parameterization," *Computers & Structures*, vol. 88, no. 1–2, pp. 120–133, 2010.

[49] T. E. Bruns and D. A. Tortorelli, "Topology optimization of non-linear elastic structures and compliant mechanisms," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 26–27, pp. 3443–3459, 2001.

Structural and Multidisciplinary Optimization: Vol. 47, pp. 49-61, 2013