Soft Comput
DOI 10.1007/s00500-015-1845-8

METHODOLOGIES AND APPLICATION

# Topology optimization of compliant structures and mechanisms using constructive solid geometry for 2-d and 3-d applications

**Anmol Pandey**[1] · **Rituparna Datta**[1] · **Bishakh Bhattacharya**[1]



**Abstract** This research focuses on the establishment of a constructive solid geometry-based topology optimization (CSG-TOM) technique for the design of compliant structure and mechanism. The novelty of the method lies in handling voids, non-design constraints, and irregular boundary shapes of the design domain, which are critical for any structural optimization. One of the most popular models of multi-objective genetic algorithm, non-dominated sorting genetic algorithm is used as the optimization tool due to its ample applicability in a wide variety of problems and flexibility in providing non-dominated solutions. The CSG-TOM technique has been successfully applied for 2-D topology optimization of compliant mechanisms and subsequently extended to 3-D cases. For handling these cases, a new software framework involving optimization routine for geometry and mesh generation with FEA solver has been developed. The efficacy of the approach has been demonstrated for 2-D and 3-D geometries and also compared with state of the art techniques.

**Keywords** Structural and topology optimization · Finite element analysis (FEA) · Multi-objective genetic algorithms · Compliant structures

Communicated by V. Loia.

✉ Rituparna Datta
rituparndatta@gmail.com; rdatta@iitk.ac.in

Anmol Pandey
anmolp2016@email.iimcal.ac.in

Bishakh Bhattacharya
bishakh@iitk.ac.in

[1] Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur, Uttar Pradesh, India

## Abbreviations and symbols

| | |
|---|---|
| CSG-TOM | Constructive solid geometry-based topology optimization methods |
| SIMP | Solid isotropic material with penalization |
| ESO | Evolutionary structural optimization |
| MOGA | Multi-objective genetic algorithm |
| SBX | Simulated binary crossover |
| ToPy | Topology optimization using python |
| $\boldsymbol{n}$ | Total nodes |
| $\boldsymbol{m}$ | Variable nodes |
| $\boldsymbol{k}$ | Fixed nodes |
| $\boldsymbol{n_{void}}$ | Special nodes |
| $\boldsymbol{n_{sym}}$ | Symmetry nodes |
| $\eta_i$ | Volume fraction of topology in $i$th generation |
| $\boldsymbol{\alpha}$ | Volume correction factor |
| $\boldsymbol{\varepsilon}$ | Ratio of required volume fraction to current volume fraction |
| $W_-, W_+$ | Width selection operator |
| $\lambda$ | Symmetry condition operator |

## 1 Introduction

One of the most important objectives for any feasible structural design is the achievement of best assembly of the corresponding structural elements, while satisfying various constraints such as amount of the given material, the shape/size of the structure or both. Topology optimization represents an advanced process of structural optimization providing flexibility for considering both internal and external boundary configurations simultaneously.

Optimization of topology in the context of compliant mechanisms is broadly of two types: truss structure-based approach (Bendsoe and Kikuchi 1988) and continuous sys-

Published online: 04 September 2015

tem approach (Huang and Xie 2010), where the complete spatial distribution of material is handled. Another popular method in topology optimization is solid isotropic microstructure or material with penalization (SIMP) which is renowned for the compactness of coding. It is suggested by Bendsoe (1989). The SIMP method discretizes material domain through uniform meshing, where every element of the mesh is treated as a design variable. A sensitivity analysis determines the importance of design variables by computing the derivatives of the objective fun10on (strain energy) with respect to design variables. Similar to the ground truss approach is homogenization-based optimization (HBO) technique proposed by Nishiwaki et al. (1998). It considers the geometric parameters of a microstructure as design variables and homogenizes the properties in that microstructure. Minimum compliance problem (minimizing flexibility of the structure under given loads, subjected to a volume constraint) is a natural starting point and standard benchmark objective for topology optimization. It is one of the simplest type of design problem formulations under resource constraints (Howell 2001). Here, the optimum distribution of material is measured in terms of the overall stiffness of structure.

The concept of inefficient material removal from a structure to obtain an optimal shape or topology is used in evolutionary structural optimization (ESO), first proposed by Xie and Steven (1993, 1994). A binary representation of design variables for solid-void elements makes it possible to produce a black and white (solid-void) optimal topology that excludes any gray (i.e., fuzzy or intermediate density) regions without using any filtering technique. This procedure is later extended by the same authors considering frequency constraints (Xie and Steven 1996). A bi-directional evolutionary structural optimization (BESO) (Young et al. 1999; Querin et al. 2000) allows for addition as well as subtraction of elements from the finite element model simultaneously. Some more works related to structural optimization are available in (Jakiela et al. 2000; Wang and Tai 2005; Kudikala et al. 2009; Tavakoli 2014). These methods, however, have few drawbacks in terms of development of checkerboard patterns and point flexures inside the design domain leading to configurations that may be impractical. Furthermore, the local optimal solutions can be obtained due to gradient-based optimization routines and may have problems due to fixed mesh formulation as discussed by Rozvany et al. (2005); Sigmund (2011). In addition, excessive computational power required due to high number of design variables often acts as a major hindrance. Ahmed et al. have addressed these issues (2012) (Ahmed et al. 2013; Ahmed 2012) and proposed a better method which comprises placement of nodes and primitives capable of formation of topology with lesser number of design variables. Through the use of population-based global optimization algorithm, NSGA-II (Deb et al. 2002), the problem of trapping in local optimal solutions can be avoided. Also, through adaptive meshing the problem of fixed meshing can be subdued (Sigmund 2011). Another study by Hamza and Saitou (2004) presented an evolutionary multi-objective optimization study based on three-dimensional geometry represented via constructive solid geometry (CSG). This study extended NSGA-II for binary tree chromosomes with customized crossover and mutation operators.

A popular and simplistic yet powerful, technique proposed by Braid and Lang (1974) allows for creation of complex surfaces and objects by the use of Boolean operators or primitives. The instance-based primitives undergo transformations as in scaling, rotation, translation and finally set-theoretic operations of Euler operators like union, intersection and difference. It offers several advantages like easy and faster construction of solid model, conversion to boundary representation and assurance of valid model.

The proposed methodology uses solid modeling techniques as they provide an excellent way for representation and visualization of topologies. Here, the topology is defined as the complete, valid and unambiguous representation of physical objects. The completeness implies that the object has well-defined boundaries and finite size. Validity of the model implies that the essential characteristics that define geometry like vertices, faces, edges are all connected properly. Unambiguity refers to the one-to-one mapping that is present between representation and actual geometry formed. The present work is also extended to handle voids, both active and passive for handling 2-D and 3-D geometries, width limit detection and improved volume penalization operator. The efficiency of our method is compared with popular SIMP methodology (Xie and Steven 1994) applied on benchmark problems. The width limit operator for deciding the upper and lower limits for the widths has been introduced to tackle the issue of convergence and providing accurate and problem-specific width limits. The volume penalization operator tries to satisfy volume fraction constraints in a single iteration of topology formation. For handling the imposed symmetry conditions on the geometry the concept of symmetry nodes has been introduced.

The paper is organized as follows. Sections 2 and 3 describe the methodology for two-dimensional and three-dimensional geometries. Section 4 presents the results of optimization. Finally, research towards future direction is discussed with conclusions.

## 2 Methodology for two-dimensional geometries

The basic methodology involved in the proposed approach comprises of utilizing a small set of points (nodes) and their respective interconnectivity through rectangular primitives to get compliant mechanisms or structures. Thus, the complete

mechanism is composed of overlapping material segments connected at different nodes.

The initial node locations are determined according to the type of the node. Every node serves a particular purpose, for example, the fixed nodes tend to define the characteristics of the boundary conditions whereas special nodes (design constraint nodes) only define the active or passive constraints inside the design domain.

The complete topology can be defined by a set of nodes and widths represented by $\boldsymbol{n}$ and $\boldsymbol{W}$, respectively. The total number of nodes, $\boldsymbol{n}$ can be divided into four categories:

1. *Variable nodes* ($\boldsymbol{m}$) Consisting of nodes which can position themselves anywhere in the design domain. These are design variables, where the coordinates are dependent on domain shape and size.
2. *Fixed nodes* ($\boldsymbol{k}$) Consisting of nodes that remain fixed in the design domain, and are used for applying boundary conditions and loads. This is similar to the constraint applied by SIMP on certain locations, for example, density ($\rho$) $> 0$ for the element where force is applied. The utility of $k$ fixed nodes lies in saving the repair operation step in making the topology comply with predefined boundary conditions.
3. *Symmetry nodes* ($\boldsymbol{n_{sym}}$) These are present when the designer wants to impose symmetry condition on the domain by specifying a line of symmetry. $n_{sym}$ defines the number of points on the symmetry line of the mechanism, where only single degree of freedom exists in terms of position.
4. *Special nodes or void or design constraint nodes* ($\boldsymbol{n_{void}}$) To take into consideration active and passive voids during iterations, we approximate the outer shape of these voids (which are obviously inside the design domain) by a set of nodes, $n_{void}$. Here, $\boldsymbol{n_{void}}$ can be interpreted in two ways: (a) A set of points which when connected would define a polygon, approximating the required outer boundary of void, or (b) A set of representative points on boundary of void, which are sufficient to actually give the actual shape of void. For example, a circle represented by three or more points on it. The circumcircle property is used to get the circle back from this set of points. It depends upon the problem and user to predefine, how the $\boldsymbol{n_{void}}$ points are to be used.
   In the present work, $\boldsymbol{n_{void}}$ is essentially a structure with arrays containing information about these voids or non-design entities, the interpretation of these arrays to get geometric output is explained in later section.

The active voids are regions where the material is absent from $\Omega$, right from start of optimization procedure. Thus, $\boldsymbol{n_{void}}$ points lying on boundary of voids make sure that the connectivity of domain is not dependent on nodes lying inside the region. The material around the void should be removable, allowing flexibility for void boundary change or elongation. To facilitate the changes in void boundaries, the widths between $\boldsymbol{n_{void}}$ nodes are treated as normal width design variables, which are changed during optimization procedure. The internal connections among $\boldsymbol{n_{void}}$ nodes would be absent or have negative widths for no material inside void.

However, in some cases material may be allowed to be present around the void. This conserves the entire or part of the shape of the void throughout the optimization process. Hence, certain widths among $\boldsymbol{n_{void}}$ nodes are fixed on the void boundary, thus removing them from the width design variable.

The passive void material constraint can be represented in a similar manner. As connectivity of material is not an issue when the material is added to the already connected geometry, the passive void constraint can be represented by a single internal node, where later through CSG union, the exact shape can be added. However, this is generally discouraged because in such cases the non-design constraint shape and size become independent or rather invisible to the optimization procedure.

### 2.1 Optimization formulation

Defining design domain ($\Omega$) completes with voids and outer boundary.

### 2.2 Objectives

Single objective optimization formulation is represented in Table 1. For multi-objective optimization, user can choose from 2 or more objectives from Table 1.

The values of these objectives are taken to be fitness values, later used by our evolutionary algorithm to keep the best geometries and eliminate the rest.

**Table 1** Single objective problem formulation

| S. no. | Objective | Equation |
|---|---|---|
| 1 | Minimize compliance | $f(x) = \frac{\min}{\rho} \int^{\frac{1}{2}} \sigma \varepsilon d\Omega$, where $\rho \in [0, 1]$ |
| 2 | Minimize maximum stress | $f(x) = \min(\text{Max}\sigma(\xi))$, where $\sigma$ can be principal, von-mises stress, etc. and $\xi \in \Omega$ |
| 3 | Minimize average stress | $f(x) = \min \int_\Omega \left(\frac{\sigma - \sigma_a}{\sigma_a}\right)^2 d\Omega$, where $\sigma$ is maximum stress and $\sigma_a$ is average stress |
| 4 | Minimize weight | $f(x) = \min(\int_\Omega \rho d\Omega)$ |

In our case, the value of density remains constant throughout the elements after discretization for FEA, i.e., $[\rho = 1]$ . Rest of the equation remains the same.

### 2.3 Constraints

One or more constraints can be taken similar to above objectives, where a predefined value given by user gives us the constraint equality or inequality equation.

For example, weight constraint condition after discretization (through meshing) done for FEA

$\sum_1^N V_e \rho_e \leq f V_{\text{total}}$, where $f$ is the predefined volume fraction constraint provided by the user.

$$\Rightarrow \sum_1^N V_e \leq f V_{\text{total}} \quad \text{as } \rho_e = 1 \quad \text{for all } \Omega \text{ where,}$$

$V_e$, volume of a single bar; $N$, number of existing bars inside the region between all nodes in $\Omega$; $V_{\text{total}}$, total initial volume of design domain ($\Omega$).

### 2.4 Design variables

The design variable array for NSGA-II (Deb et al. 2002) is taken as [*width variables, location variables*]

Number of nodes: $n = m + k + n_{sym} + n_{void}$, where $n$, total number of nodes; $m$, total number of variable nodes; $k$, fixed nodes, used for boundary conditions; $n_{sym}$, nodes for enforcing symmetry; $n_{void}$ nodes for void location and shape definition.

Number of node location variables for 2-D topology, $N_{(x,y)} = 2 * m + \lambda * n_{sym}$,

$\lambda = 1$, if symmetry condition is applied, else $\lambda = 0$

Number of width variables:

$$w = {}_2^nC - \sum_{i=1}^{N_{voids}} {}_2^jC + \beta * j$$

$N_{voids}$, number of active voids present; $j$, number of sides in $i$th void; $\beta = 0$, if void boundaries are user defined, else $\beta = 1$

Total number of design variables, $n_{design} = N_{(x,y)} + w$

$$N_{(x,y)} \in [0, 1] \quad \text{and} \quad w \in \left[W_-, W_+\right]$$

$[W_-, W_+]$: Limits given by the width selection operator.

### 2.5 Algorithm summary

The algorithm for topology generation can be summarized in the following steps:

*Step 1*. Define total $n$ nodes with $m$ variable nodes and $k$ fixed nodes denoting the boundary conditions, $n_{fix}$ denotes voids or non-design constraints and $n_{sym}$ denotes symmetry line points.
*Step 2*. Interpret variables to obtain actual positions of nodes in the design domain.
*Step 3*. Use constrained Delaunay triangulation to obtain connectivity between all nodes.
*Step 4*. For the first topology, constant width allotment step is used for determination of the lower and upper bound of width.
*Step 5*. Interpret genes to obtain the widths of bars corresponding to edges in allowed connectivity set with only positive values to be considered.
*Step 6*. Check topology consistency through graph-based repair method.
*Step 7*. Use CSG tool to obtain the topology after union of all bars. Apply symmetry condition if required. Also carry out CSG removal of voids and CSG addition of non-design constraints.
*Step 8*. Mesh the obtained topology using CSG-supported mesh generator.
*Step 9*. Apply loads and boundary conditions at required nodes.
*Step 10*. Use of finite element analysis to solve for deflections and stresses.
*Step 11*. Calculate volume fraction of present topology and apply volume penalization function if the output is greater than required.
*Step 12*. Based on required output, return fitness and constraint values are calculated. Next generation of output geometries is generated by NSGA-II, and we go to Step 2.

### 2.6 Defining and interpreting nodes

The boundary conditions and loads are user defined. The locations and shape of voids and non-design regions are defined in the form of representative nodes, $n_{void}$. The location of the variable nodes is parameterized (between 0 and 1) and when obtained as design variables from the optimization procedure have to be interpreted to get the Cartesian or Spherical Coordinates. These then provide locations where material can be present or absent depending on the type of node.

The distribution of variable nodes $m$ inside the domain has to be defined considering active voids as passive ones which define the non-design constraint region boundaries, so that they never lie inside these voids. To ensure this, we interpret the actual location of the nodes, for example, multiplying by maximum $x$-coordinate of domain. The Cartesian $x$-coordinate is thus obtained as $x_i$ and then to get the $y$-coordinate, $y_i$, we uniformly distribute the parameterized $y$-coordinate from location design variable, along the $x = x_i$

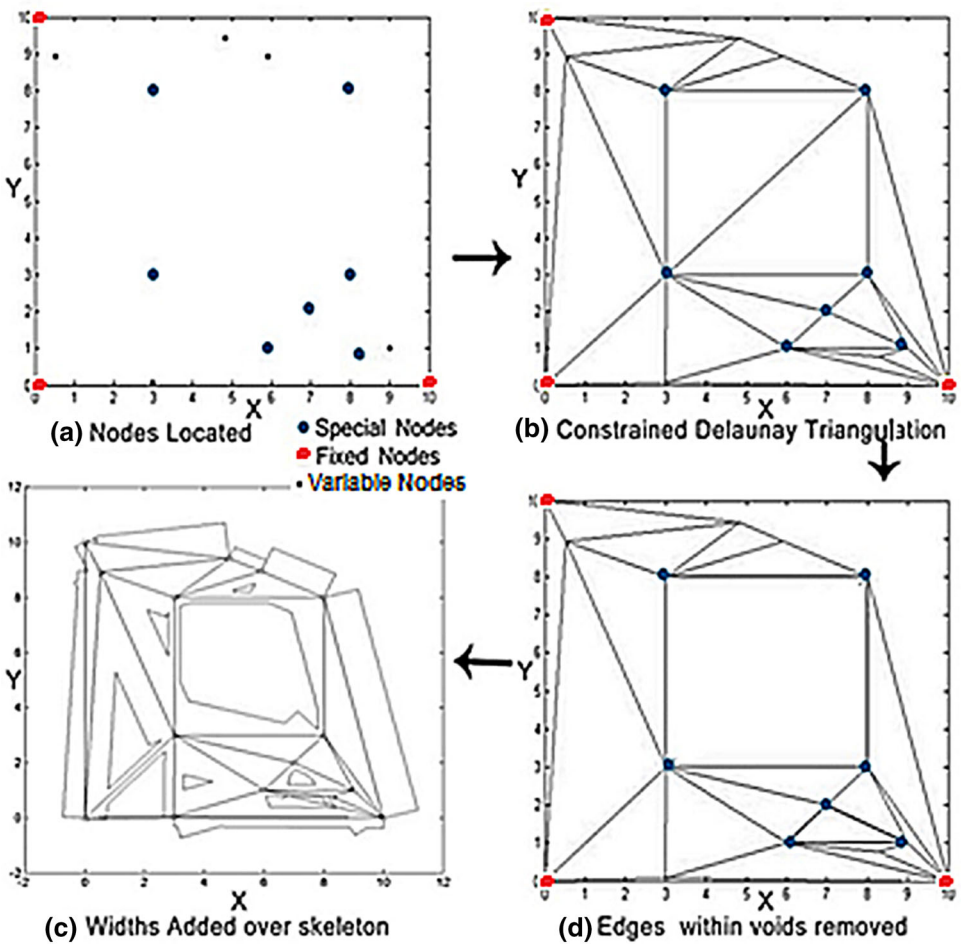



**Fig. 1** **a** Node placement of different types within the design domain, **b** constrained Delaunay triangulation with edges forming the outer shape of void, **c** edges lying inside the voids are removed, **d** *bars* between node connections are added over triangulation according to the width design variables (the *arrow* directions point towards the following step)

line bounded by the design domain. The regions where the constraint (mainly voids) encounters the line are skipped. The node placement is explained in Fig. 1a, where $n_{fix}$ have been clearly defined lying on the vertices of rectangular and triangular voids.

### 2.7 Nodal connectivity by triangulation

In this step, we find the underlying connectivity between the nodes over which the primitives would then be placed. The reason for choosing Delaunay triangulation (Lee and Schachter 1980) is mainly to avoid inter-connections between every node to another, thereby leading to ${}^{n}_{2}C$ connections, making it extremely computationally intensive with large number of intersecting bars. Also, there will be a lot of artificial node points created due to beam intersections, which makes it for the optimization procedure to eliminate the non-required widths. The geometry thus formed will be subjected to excessive overlaps and high volume fractions.

To avoid these problems, 'Delaunay Triangulation' comes in handy. It decreases the number of design variables in the form of widths which would actually be used, leading to less overlaps and artificial node formations. This decreases the complexity in geometry formation and boundary information extraction through *decsg* function of MATLAB (Guide 1998; Mathworks 2013).

Geometrical constraints put on Delaunay triangulation method help to trace the outer boundaries of voids in the shape of the polygonal vertices as illustrated in Fig. 1b. It always defines nodal connection outside of voids, which in turn helps in identification and integration of void presence in the algorithm itself so that artificial material removal through CSG subtraction in step 7 is avoided. This ensures that geometry defined by variables undergoes minor changes, thus minor changes in fitness values after Step 7.

The set of edges obtained by applying this method between the nodes can be called as *Base Skeleton* of the topology. It provides an easy way of visualization of node connections. The internal set of edges inside the voids has to be eliminated to give us the final skeleton over which in the next step we would form the body as shown in Fig. 1c.

### 2.8 Bar allotment in CSG

After obtaining the skeleton, the edges have to be replaced by rectangular bars whose widths are passed on to us as design variables by the optimization procedure. The boundaries around active voids if user defined would not be the part of width design variables.

Care has to be taken that for special nodes representative of constraint regions, the user provided widths are placed at the correct positions, for example, for connections lying inside the voids, width value has to be zero as shown in Fig. 1d.

The reason for not choosing the primitives different from rectangles or cuboids is that it is the simplest shape capable of defining the connections between a pair of nodes. Choosing any other shape would always lead to increase in number of design variables and thereby making it harder to find optimized pareto-front solutions. However, one problem arising out of this would be stress concentration around the ends of rectangular primitives.

Taking round edges at the end of rectangular primitives for stress concentration alleviation does not solve this problem due to two reasons. First, the sharp corners would still be generated; as on every node through CSG union various intersecting elements are joined and they cut each other at various angles. Hence, the final topology will still have the same problems as with rectangular bars. Secondly, as the width decreases the tolerance level of CSG union has to be decreased to take into account various small radii present at the end of each bar. This in turn adds to the complexity and time in getting the final topology from union of all elements.

The solution would work well if all the bars had same width, thereby leading to formation of a circular shape around the node. But that puts up the restriction in the range of solution one can get out of combination of design variables.

For the cases where one of the objectives is taken as the von-mises stress value, it is taken to be the average stress occurring in the entire topology. This leads us to eliminate the geometries with higher stress concentration regions. Also, this ensures that any geometry is not eliminated due to one peak occurring because of stress concentration, which would be the case if objective function is taken as max stress value. Generally, the CSG Union of rectangular bars creates such regions. These regions must be handled during post-processing of geometries, for example, one way of reducing sharp edges, and thereby the stress around the nodes is by placement of small circles over them during CSG union.

### 2.9 Symmetry condition

The handling of symmetry conditions within the design domain is sometimes essential for the topology. However, with Delaunay triangulation, the achievement of triangulation is difficult because the triangulation itself does not support symmetry, even though the elimination of widths and nodes might align and after certain generations, converge towards the intended symmetry.

The problem can be solved by enforcing the symmetry required by designer along a line. This is achieved by limiting the variable nodes to one side of the symmetry line, forming the base skeleton and adding primitives. The symmetry condition is used to mirror the primitives on other side. This seems to be simple, however, the problem lies in interconnectivity of the two regions as illustrated in Fig. 2. If there are points on the line of symmetry, then it is ensured, but this might not be the case. To alleviate this problem, we include another set of nodes, $n_{sym}$ which are just like variable nodes, but have only one degree of freedom to move along the line of symmetry, hence represented by a single design variable. This ensures that symmetry as well as connectivity issues arising out of enforcing it is addressed simultaneously.

### 2.10 Topology repair operator

This technique is introduced to handle any inconsistencies in topology which might arise due to lack of interconnectivity caused due to certain width variables having zero values. This would lead to failure in FEA analysis and give an error, ending the entire process. Hence, detection and corrective action before the FEA step is necessary. Rejecting or penalizing the geometries would further waste function evaluations and would waste the space of one population member with valid topology. The disconnected geometry can be formed in the following ways:

1. The point at which load is applied and boundary conditions for the nodes are disconnected.
2. Nodes representing non-design constrained regions.
3. Presence of non-connected materials in the topology.

The algorithm for repair operator is explained in “Appendix 1” and briefly illustrated through an example in Fig. 3.

The repair operation seems artificial as topology is changed after getting the design variables from the optimization process, but at the same time it is vital for getting the connected geometry among all the nodes. The modified design variables then replace the original ones thus promoting only those population members where the repair operation is not needed at all.

The voids where the external boundaries are not given by the designer are not included in the ’Special Set’ so that flexibility is available for elimination of more regions around void if required and the void is not always bounded by material.

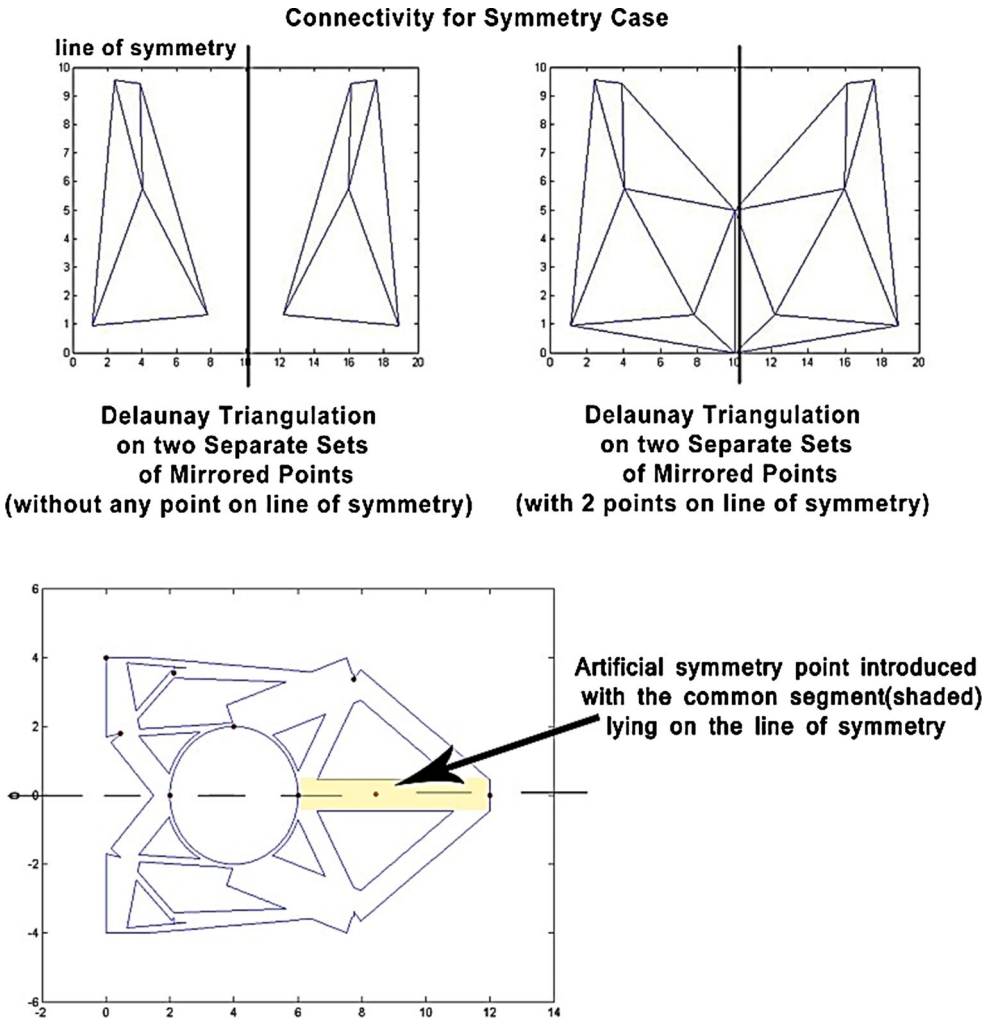



**Fig. 2** Illustration explaining the importance and need of 'Symmetry Points' for handling symmetry conditions within given design domain

### 2.11 Width limit detection

At the start of the optimization procedure, the width limits, $\left[W_-, W_+\right]$, have to be decided and specified for width design variables. The limits stretch from a negative to a positive value to decide the interconnectivity among the nodes. For a positive value, the node connection is present in the triangulation and for negative value it is absent. The negative width actually gives power to the optimization code to remove unwanted segments. Initialization and further functionality of width variables inside the width limits both lower and upper, not only decides the probability $p_{width}$ of number of segments definitely present in the geometry, but also contributes towards convergence of the algorithm. It can be observed that if the probability is made too small for negative widths then elimination of certain nodes from the geometry might not at all be possible. The possibility of eliminating certain unwanted nodes helps in formation of simpler geometries which might be the solution. On the other hand, if positive width probability is kept low then all the solutions might find simpler geometries, through topology repair operator.

At this point, it should be noted that MOGA is capable enough of dealing with these probabilities and moving towards a global maxima. However, the parameters to handle them might imply more time or generations for convergence to pareto-optimal front. Also, it might need a larger population size. Here, the term 'Global Maxima' obviously is dependent on the number of nodes. So for creating a sense of balance we take the probabilities to be 50 % ($p_{width} = 0.5$) or the lower and upper limit having the same numerical value.

The positive width limit is always decided from the width limit operator and the negative width limit is decided accordingly to attain the required value of probability. If $p_{width} < 0.5$, i.e., the negative width becomes greater than the positive it may lead to greater than required widths. To avoid this

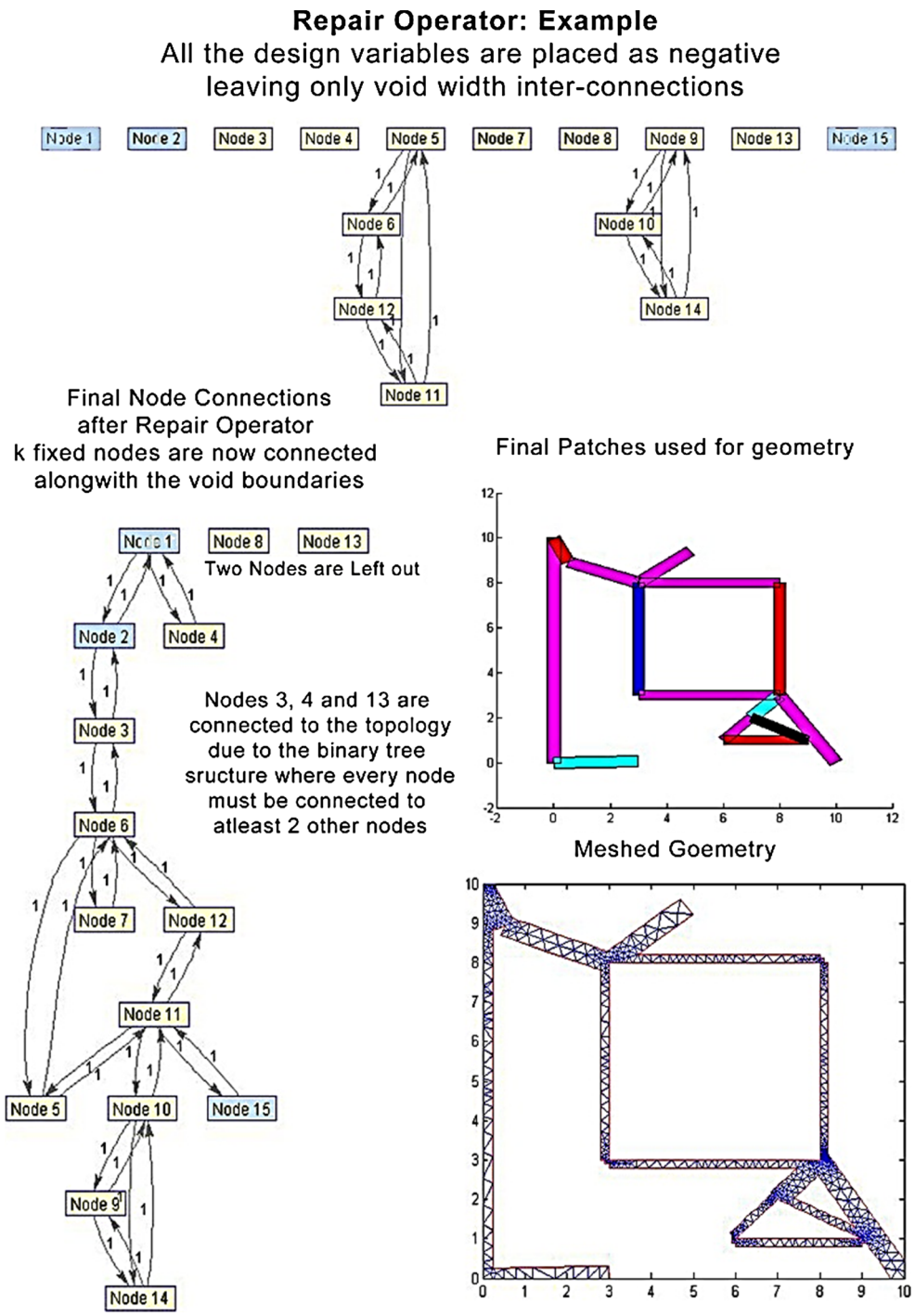



**Fig. 3** An example of repair operation for obtaining connected and valid geometries

condition when applying the repair operator, if the numerical value of the negative width becomes greater than the numerical value of the positive width, then it is made equal to the positive width limit. This ensures that the optimization functions like crossover and mutation remain unaffected and work in the full range $\{W_-, W_+\}$ whereas, in repair operator the widths remain equal or smaller than the actual positive width limit decided by the width operator.

If the design domain is changed in size or the number of nodes is altered, the width limits have to be accommodated accordingly. This is done through the following steps:

1. For the first topology, we take constant positive widths for all segments, starting from a low value, example 0.5.
2. This constant width is increased in predefined steps.
3. The topology is formed and volume fraction is calculated.
4. If the current volume fraction for $i$th iteration, $\eta_i$ is greater than 0.9 or the change from previous iteration is less than 5 %, then we limit the numerical width limit as two times

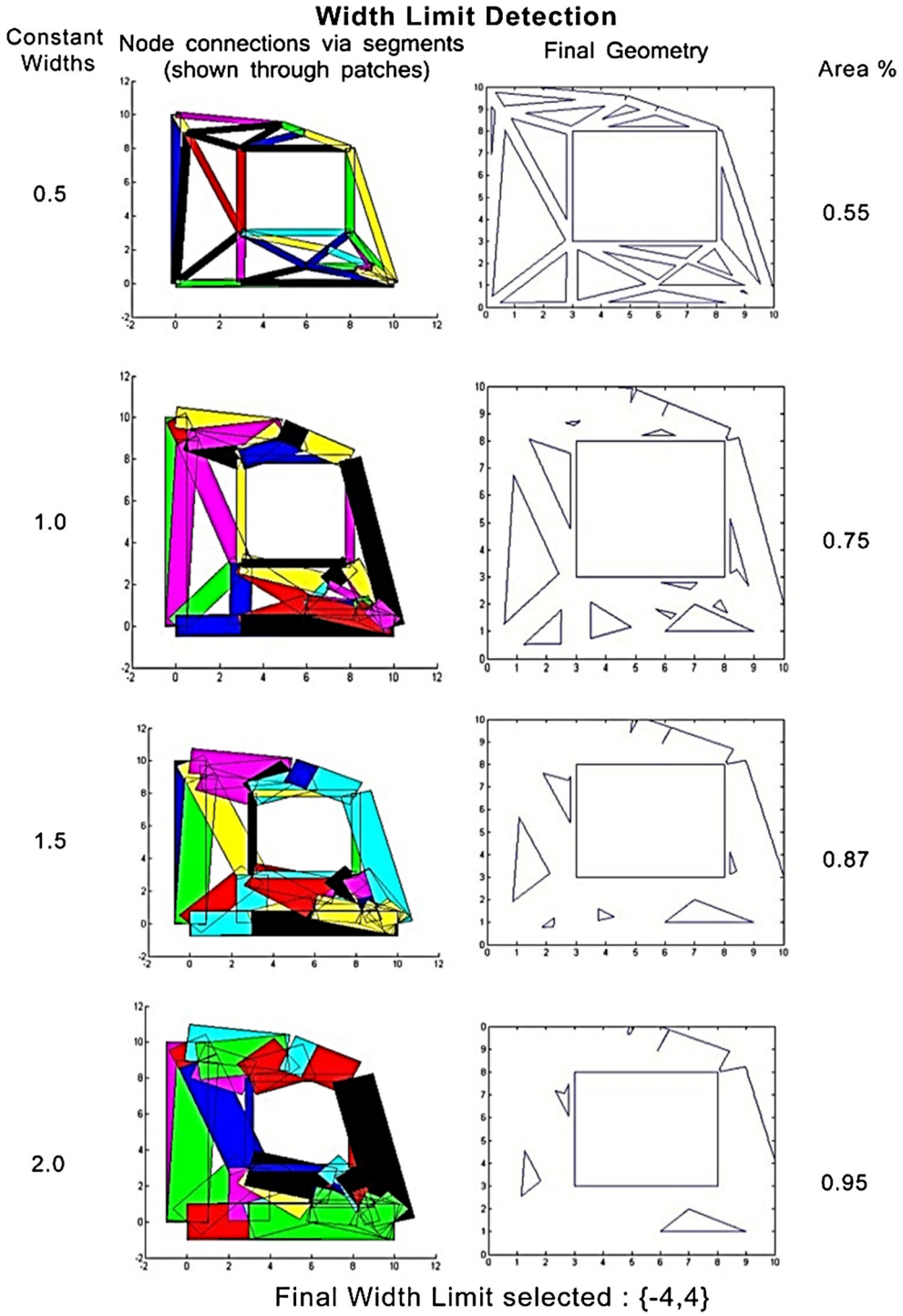



**Fig. 4** Width limit is decided based upon the percentage of area, the maximum width allotment can cover the design domain (different segments defining the topology are shown in *different colors*) (color figure online)

that of the current constant width, $W_i$. Otherwise, go back to Step 2.

5. Calculate the respective positive or negative width limits as:
   Positive width limit, $W_+ = 2 * W_i$
   Negative width limit, $W_- = -[W_+ * (1 - p_{width})]$.

The iterations for different widths are shown in Fig. 4 until the final width is obtained. The $W_+$ is kept twice as all the segments with maximum width limits (a rare case) would certainly not be present. This increases the range of volume fractions that can be covered by the width limits. This best works for requirements of geometries with volume fractions below 0.7 and a yet higher multiplication factor to $W_+$ would further increase this range. If at the end of iterations $\eta_{final} < f$ (required volume fraction) implying that most of the topology cannot be covered or approximated by these particular set or number of nodes; in such cases, the number of nodes is to be increased to cover more regions and the algorithm is run again to ascertain new limits.

If the required optimization has put constraints on the volume fraction for the final topology, it would be better if this information is in some way integrated in the process of definition of width limits itself. This is done by replacing limit on current volume fraction in Step 4 by $\eta_i = 1.5 * f$. A smaller range of widths for primitives helps in satisfaction of the volume fraction constraint by generating topologies with volume fractions around $f$ itself. This process surely helps for faster convergence as a greater range poses an unnecessary challenge for MOGA to overcome. It also reduces the number of initial population members required to explore and solve the problem.

The widths that are too small in size to be manufacturable/machined have to be removed or repaired in a way that interconnectivity is still maintained. So the tolerance value, $\delta$ (taken as 0.1) for widths is decided and any value lower than that is brought up to the value $\delta$ helping in elimination of poor and impractical linkages.

The parametric study in our case would primarily involve types and number of nodes taken and other is the shape of the bars. Number and type of nodes to be taken for boundary condition definition or active/passive design constraint depend on the shape and condition that is applied. Different boundary conditions have been shown in various examples.

The need for width limit detection itself came from the difficulty in identification of correct parameters required for variable nodes. For the number of variable nodes, the width limit detection ensures that it is formulated beforehand and is sufficient enough for covering the entire topology. If the variable nodes are not enough then the maximum design domain would not be covered even by increasing the range of widths. The operator in the beginning itself performs a parametric study to determine the nodes ideal for representing the topology.

### 2.12 Volume penalization operator

The minimization or limitation of volume/ weight is one of the most common and important objectives in topology optimization (Sokolowski and Zolesio 1992). For single or multi-objective problems where finding different topologies for different volume fractions is one of the objective, this volume penalization operator need not be used. This example has a constraint of 50 % volume utilization ($f = 0.5$),

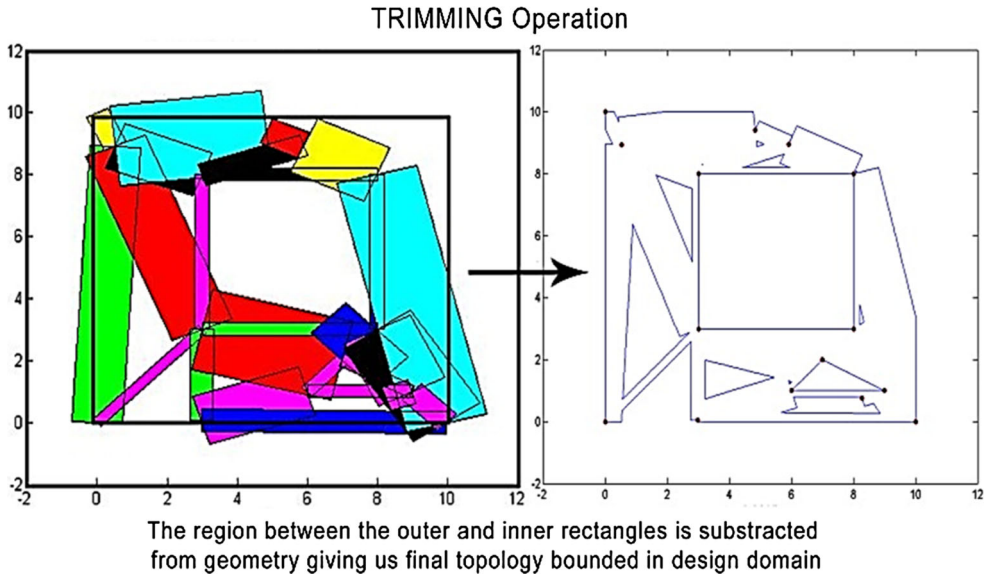



**Fig. 5** Trimming operation to get the final geometry inside the design domain, deleting the outside parts through CSG operations

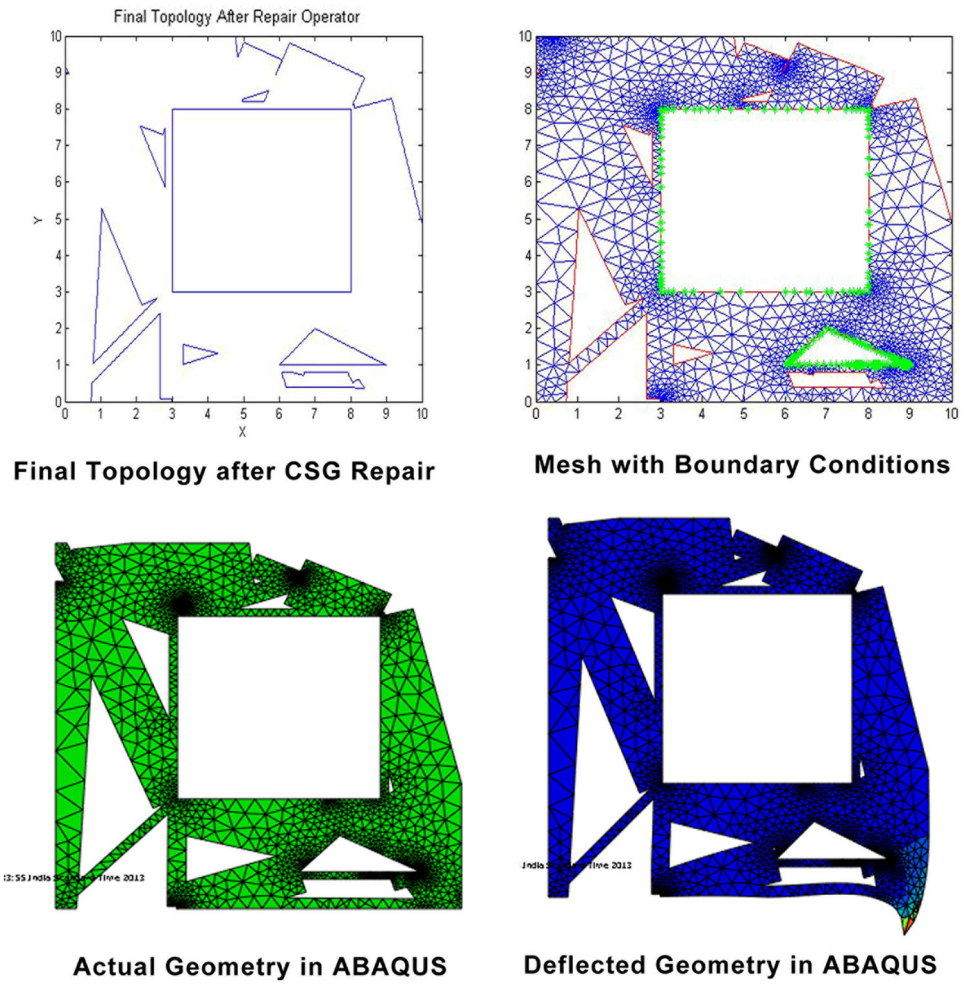



**Fig. 6** Final geometry formation with meshing done on it. The actual and deflected geometries after analysis in Abaqus are shown

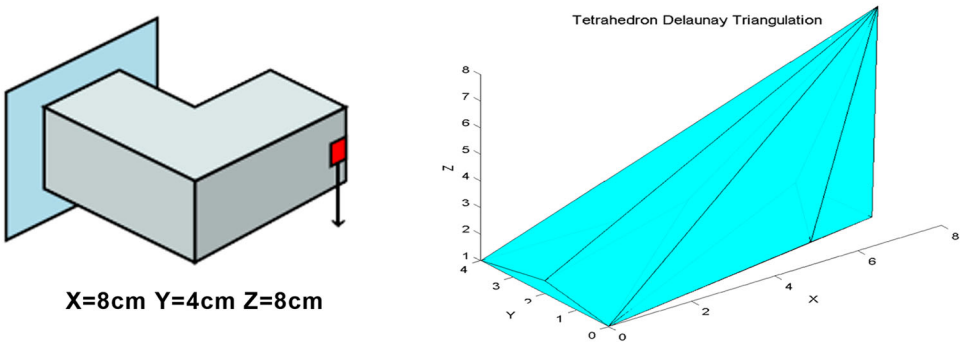



**Fig. 7** 3-D delaunay triangulation

where topologies using above this limit would be deemed infeasible.

Without any particular estimate of the amount of overlapping that is taking place among the segments at the nodes, the exact change in width required cannot be determined. Previously in Cuillière et al. (2013), the correction operator to avoid any infeasible topologies would calculate the volumes and then for those showing constraint violation multiplies their current widths with the ratio,

$$\varepsilon = \frac{f}{\eta_i} \tag{2}$$

where $\eta_i i s$ the volume fraction for the topology in $i$ thiteration. $\varepsilon$ denotes the violation of the set volume fraction limit. Larger the violation, less is the value of $\varepsilon$. With the information about the extent of violation of constraint, subsequently the new volume is calculated. This process is repeated until the constraint violation is eliminated.

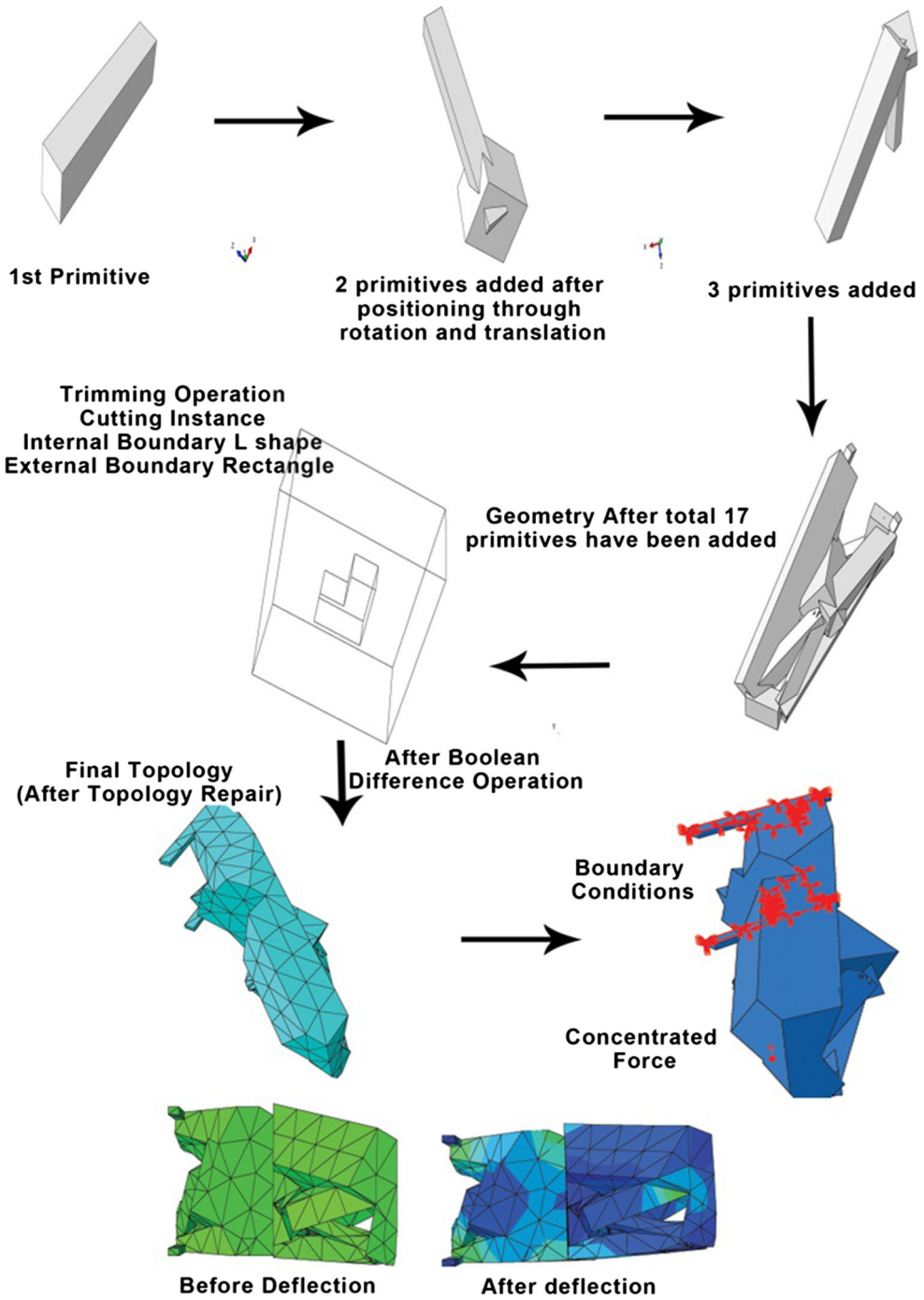



**Fig. 8** Process of 3-D topology formation and analysis inside FEA Solver

The process although, always favorable for constraint satisfaction, still takes up valuable time in terms of multiple times geometry formation. A need is felt for an operator which is able to do this in a single iteration. The ratio $\varepsilon$ which was previously multiplied to the widths of all segments is now replaced by $\boldsymbol{\alpha}$ to account and correct the severity of volume fraction violation in a single iteration.

$\alpha = \min[\varepsilon^{(1+\varepsilon)}, 0.9]$ where new width variables $W_{new}$, in case of violation become $W_{new} = W/\alpha$.

The ceiling for the value of $\alpha$ is necessary to make the small constraint violations ($\eta$ just greater than $f$) equal to zero. As the severity of constraint violation increases, the original widths are reduced more.

$W_{new}$ replaces $W$ in the parent population, thereby making sure the solutions have least possible violation of volume

fraction constraint. This helps in converging towards geometries already satisfying the required volume constraints. The process in some cases makes the volume fraction too small and well below the limit because of raised power on $\varepsilon$ or it might fail in bringing it inside the limit. The lower volume fraction geometries can still climb up towards the volume fraction limit ($f$) in the next set of generations. The trade-off for time and function evaluation with the exactness of constraint satisfaction can easily be observed and the savings due to elimination of function evaluations wasted on the same topology make it a worthy choice.

### 2.13 CSG subtraction and addition: trimming

Even though the nodes are always kept inside the respective domain, but due to the addition of widths between them as material, it may violate the boundaries. A small portions of the geometry might go out of the bounding boxes and interfere with the void boundaries. On the other hand, the material for the non-design constrained region has to be added. So, this operation subtracts the voids along their boundary and applies CSG union for adding the required material. When the void boundaries are removed through CSG intersection then at the void boundary vertices, the condition of point flexure might occur. To solve this problem we take small rectangular patches which are then placed at all the vertices. This ensures that a segment does not have a connection only at a single point, avoiding point flexure. This condition only arises when the material is removed through CSG difference in the cases of voids.

Also, the outer boundaries have to be enforced. For that we take a bigger square with sides equal to the maximum length/width possible. Then, the internal boundary is subtracted from the bigger square. This is later subtracted from the repaired geometry of the previous step. This gives us the final geometry as in Fig. 5.

### 2.14 Meshing, FEA analysis and output

The final geometry from previous step is then triangularly meshed through MATLAB (Mathworks 2013) followed by mesh refinement if deemed necessary. This is done to improve accuracy by increase in node density inside the design domain. The boundary conditions and load nodes are found and written in the input file for the commercial FEA solver, Abaqus (2006). The final output for compliance, stresses is obtained from the result file generated at the end of the analysis and can be visualized as shown in Fig. 5.

### 2.15 Multi-objective optimization algorithm, NSGA-II

NSGA-II is one the most popular algorithms to handle multiple objectives simultaneously. NSGA-II has shown its efficacy in many structural (Abaqus 2006) and other engineering problems (Datta and Deb 2011; Deb and Datta 2012). In NSGA-II-based algorithm, the natural selection criteria decide the emergence of a better solution and the convergence criteria are set when the best solution does not change for succeeding generations (Figs. 6, 7, 8).

Sensitivity analysis-based algorithms like SIMP define sensitivity as:
$\frac{dc}{d\rho_e}$, $c$ is the objective and $\rho_e$ represent the design variables where $(e = 1 \ldots n)$, which in this case are the individual density values of the discretized blocks.

For our case, the density-based approach is not employed because the design variables defining the location and width in their own sense do not have any physical significance in terms of affecting the objective value. It is after the CSG operations that we get the final topology and objective value. So, the sensitivity-based approach does not work with the kind of variables we are using to define our problem. Only non-gradient-based optimization algorithms have to be employed. As the search for the right topology is fastened through population-based methods, the decision to go for NSGA-II was made.

The crossover operation in NSGA-II between two parent genes is meaningful if the variables represent the same entity. In the present study, the variables representing node locations ($N_{(x,y)}$) are sorted according to $x$-coordinate. The index of the width corresponding to connection between nodes $i$ and $j$ in the width variable array can be calculated as:

$$\text{Width Index} = \left(n - \frac{i}{2}\right) * (i-1) + (j - i), \quad \text{where } j > i$$

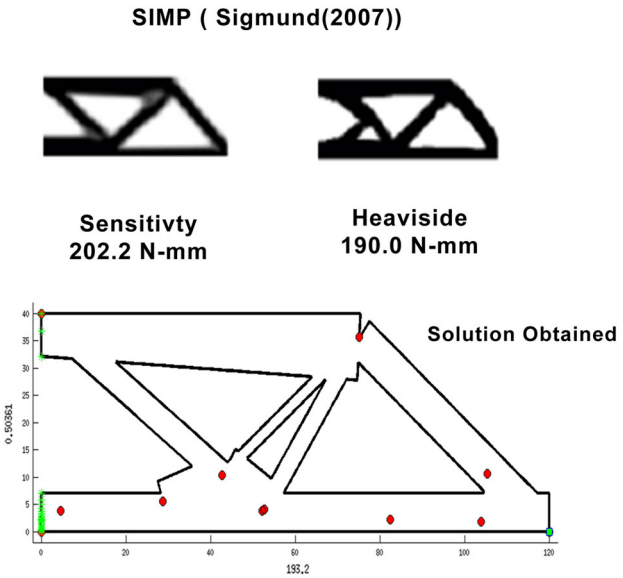



**Fig. 9** Benchmark solution and comparison

**Table 2** Problem information with relevant parameter values

| | |
|---|---|
| Number of variable nodes, $m$ | 5 |
| Number of fixed nodes, $k$ | 3 |
| Number of special nodes, $n_{void}$ | 7 |
| Number of symmetry nodes, $n_{sym}$ | 0 |
| Total number of variables, $n$ | 106 |
| Population size | 40 |
| Generations | 100 |

This ordering of variables ensures that location and width variables are crossed with their same counterpart in the population making crossover more useful.

## 3 Methodology for three-dimensional geometries

The methodology for handling of three-dimensional topologies is quite similar to that explained in the previous section for two-dimensional geometries.

### 3.1 Implementation

The formation and analysis of three-dimensional geometries warrant the use of python-based (Abaqus 2006) application programming interface, which overcomes the current limitation of CSG implementation toolbox (Mathworks 2013) in MATLAB. The scripts can be used for the entire analysis

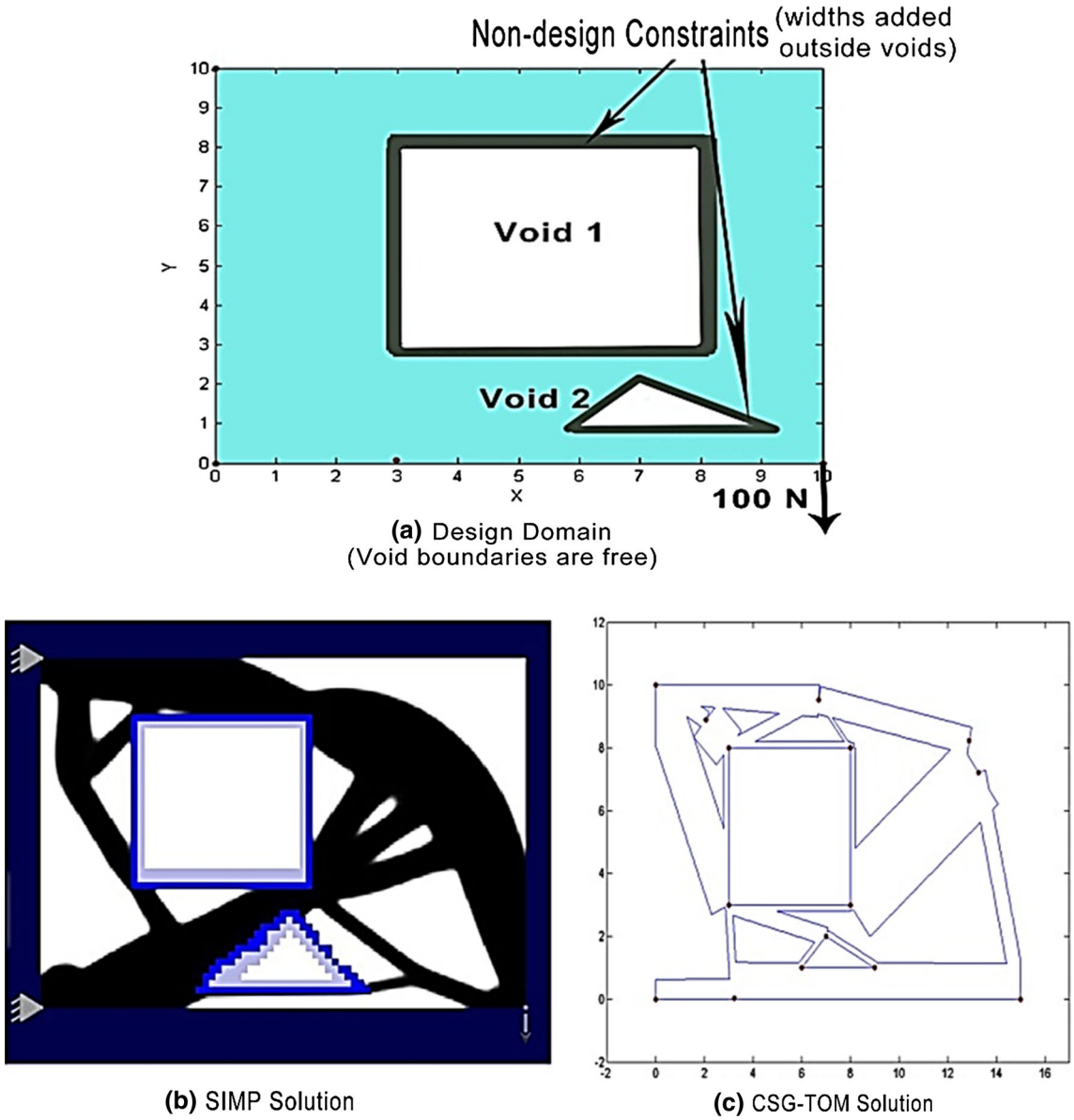



**Fig. 10** **a** Design domain with free boundary of voids. **b** SIMP methodology solution (Wang et al. 2011) along with **c** CSG-TOM solution are shown

right from topology formation to viewing results from output, which exactly serves our purpose. The optimization procedure, the geometry formation through repair operator remains the same. The interconnectivity of the nodes is decided and then processed through python-based script for primitive formation, assembly, meshing and analysis. The output of the analysis is further processed by a MATLAB routine (Braid and Lang 1974) and the objective functions are calculated and passed to the optimization routine for that particular topology.

However, the MATLAB-based implementation for two-dimensional geometries has a major advantage over Python-based implementation for three-dimensional geometries related to the volume penalization operator. For 2-D topologies since the meshing was being done by MATLAB only, the volume of any geometry could be readily obtained. This is not possible for 3-D as the geometries are being formed in the Abaqus (2006) interface itself and, therefore, have to assembled, steps for analysis defined, geometry analysis performed and from output the volume has to be obtained. Therefore, it takes more time in computation of volume fraction for constraint satisfaction.

### 3.2 Miscellaneous differences

The interconnectivity of nodes through skeleton formation between them has to be carried out through the process of tetrahedron-based Delaunay triangulation. The geometry formed due to the union of various rectangular parts, in a bounding domain, would lead certainly to a complex geometry. The geometry might comprise of multiple small edges and faces which can be eliminated to yield a simpler, more comprehensible and reliable geometry both for designer and manufacturer.

One of the other notable differences is the usage of topology repair function to handle cases where the edges are too small due to the vertices in the geometry formed being too close. This causes FEA solver to fail in the final analysis due to the in-built tolerance value in incremental steps taken to solve the problem. These cases of topologies have to be repaired by merging the two vertices.

Also, handling of the poor distorted elements is important for avoiding failures in analysis. The element type used for meshing these elements is of tetrahedron type, tetrahedron, which is seeded and completed by FEA solver itself. Now, it

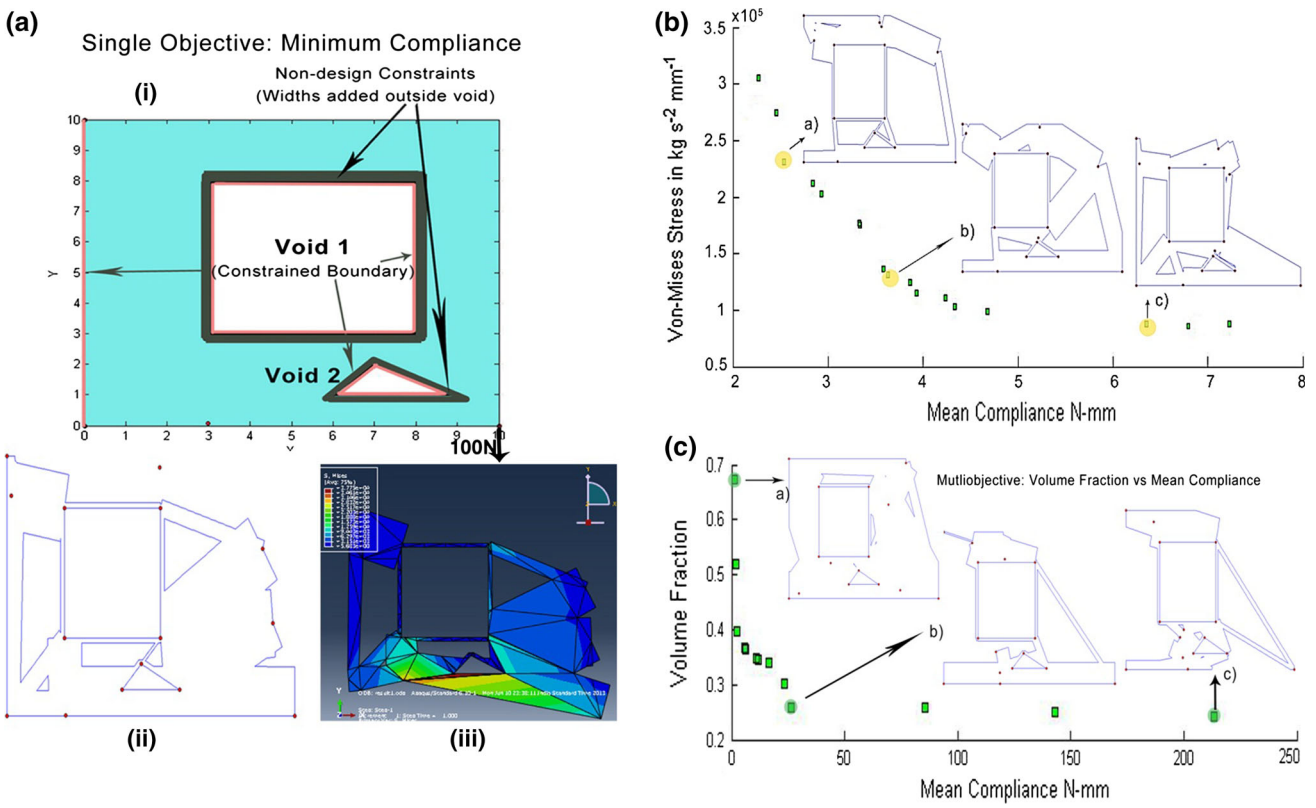



**Fig. 11** **a** *i* Design domain with fixed boundary condition around voids. *ii* Single objective solution obtained is shown, *iii* deflected solution is visualized, **b** multi-objective Pareto-Front for von-mises stress and mean compliance, **c** multi-objective pareto-optimal front for volume fraction and mean compliance

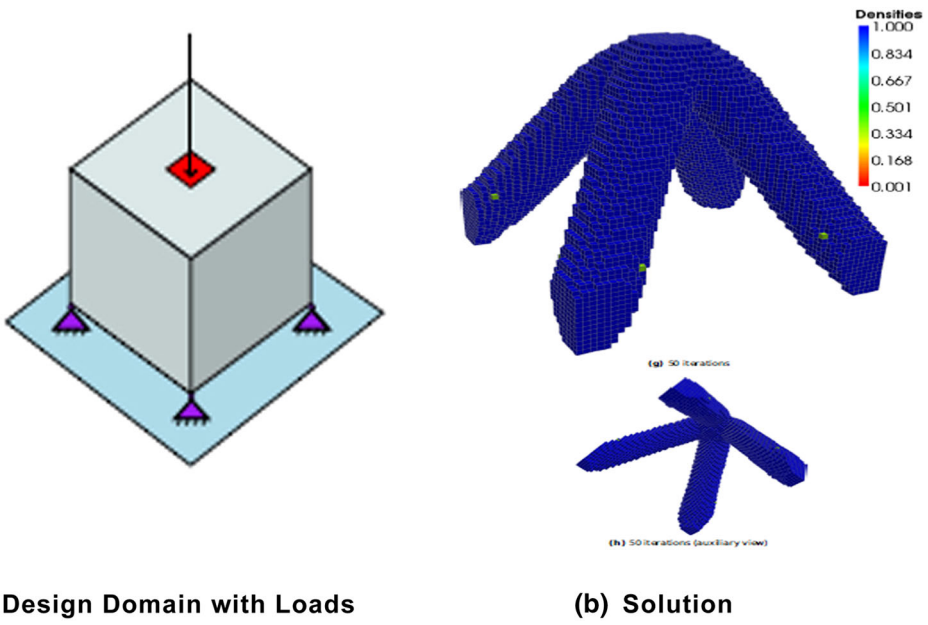



**Fig. 12** Design domain along with literature solution (Abaqus 2006)

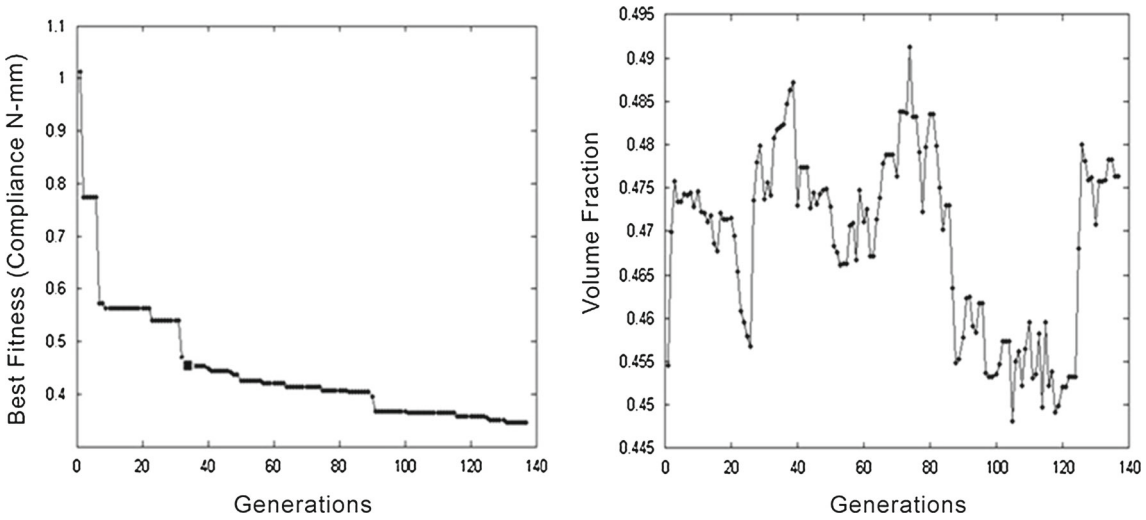



**Fig. 13** Graphs of mean compliance and volume fraction with increasing number of generations

may happen that three closely located nodes lying in a plane might form triangular planar elements. These elements then are to be assigned as 'Homogeneous Shell Section' due to absence of the third dimension.

The nodes are on the end faces of the cuboidal primitives used to join them. The node location where the boundary conditions and loads would be applied has to be found out on these faces, but the meshing operation might not be able to make the respective nodes of the tetrahedron elements on exactly the same coordinate. The partitioning of the geometry along the node is sometimes made useless by the trimming operation. To resolve this problem, the regions where the boundary conditions and loads have to be applied are added artificially in the form of small volume primitives so that the faces or vertices where these are applied always remain unchanged. As these are common among the different topologies, desirable node locations can easily be detected.

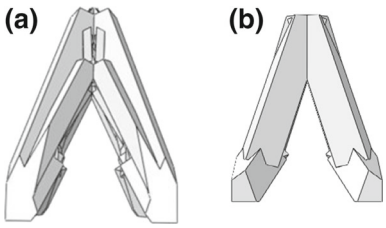



**Fig. 14** Trestle solution for single objective of mean compliance **a** solution for $m = 5$, **b** solution for $m = 2$

## 4 Results

### 4.1 Two-dimensional geometries

#### *4.1.1 Benchmark problem*

The effectiveness of CSG-TOM method was established previously by Ahmed (2012), Ahmed et al. (2013), which compared it with various sensitivity analysis-based algorithms and results showed better or comparable efficiency with these popular methodology solutions. For the rectangular domain of $(120 \times 40)\,\text{mm}^2$ (Fig. 9), a minimum compliance of 179.1 N-mm was reported. Also, for the case of symmetric loading the wheel type structure was obtained whose minimum compliance was reported to be around 9.83 N-mm.

The solution obtained for cantilever beam problem was found to be slightly better in terms of minimum compliance from the solution obtained through SIMP methodology both when using Sensitivity and Heaviside analysis (Sigmund 2007).

Upon application of our new methodology, the solutions obtained have minimum compliance of 193.2 N-mm for the cantilever beam problem when run for 150 generations. For this simple case of cantilever loading, we have used $m = 15$ and $\eta = 18$, also $n_{sym}$ and $n_{void}$ are kept zero. This makes the parameters same as used in Ahmed (2012). The small change in fitness value can be attributed to the volume penalization operator, which is forcing the geometries to satisfy volume fraction constraint in only one extra function evaluation through width reduction and subsequent replacement of original design variable with the reduced ones.

The optimization algorithm, NSGA-II parameters common to all the case studies are $p_c = 0.5, \eta_c = 15$, (probability and distribution index for SBX crossover Deb and Agrawal 1995) and $p_m = 0.01, \eta_m = 20$ (probability and distribution index for polynomial mutation).

The material data for problems are taken to be $E = 10\,\text{GPa}$, $\nu = 0.3$. The material density is taken to be $2700\,\text{kg/m}^3$ (Table 2).

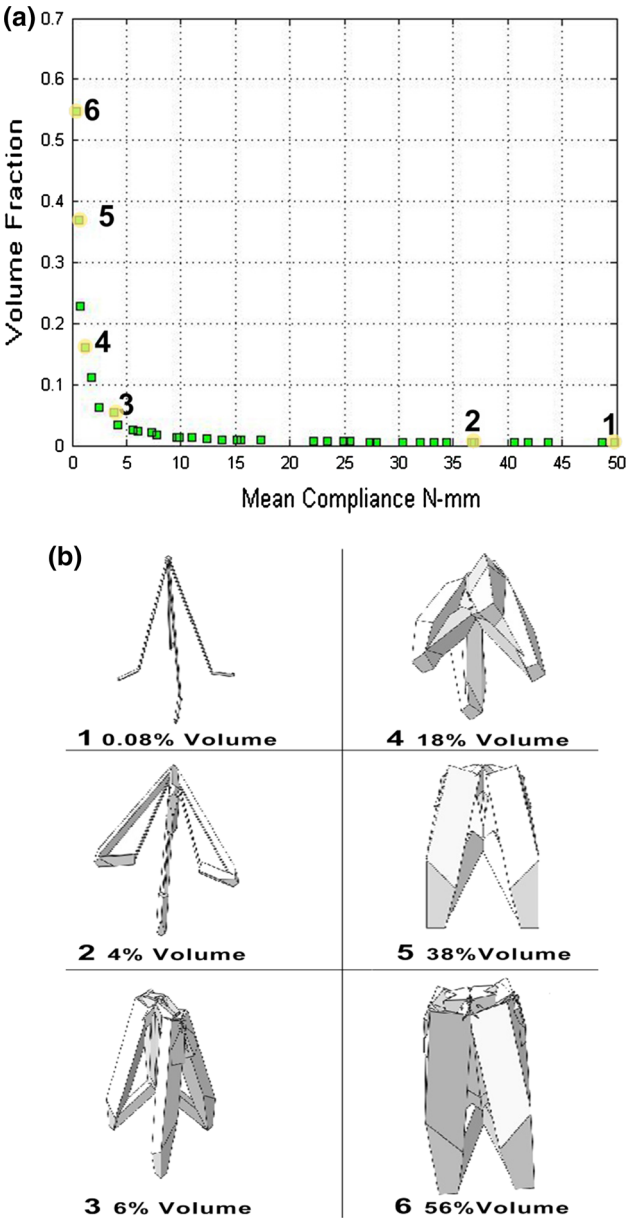



**Fig. 15** **a** Pareto-Front for multi-objective problem. *Numbers* in the figure indicate the solutions shown in (**b**). **b** Different volume fraction solutions taken from the Pareto-optimal front as shown in (**a**)

#### *4.1.2 Test problem*

This case study is divided into two parts differentiated by the boundary conditions around the two voids present inside the design domain. In the first case, the void boundaries have fixed boundary conditions, whereas in latter, void boundaries are free.

General problem information common to all studies with different objectives for Case 1 is provided in Table 1.

For free void boundaries, the solution with SIMP methodology is compared with that obtained by CSG-TOM in Fig.

10. The likeness of the two solutions is evident and clear. The CSG-TOM methodology using $m = 5$ to achieve the possible global solution, which using rectangular primitives can only represent an approximation due to curves being part of it.

Figure 11a shows different solutions obtained with varying set of objectives for the presence of fixed boundary condition on void boundaries. It implies that the material placed between fixed nodes and voids would experience minimal load influence, so the material in this domain should be minimal (just enough to keep the fixed nodes connected). Thus, material on the other side of voids should be added through node placement primarily deciding the values of mean compliance and stresses.

It is observed in Fig. 11b that a truss type structure helps to minimize the compliance whereas for minimum von-mises stress the material tends to align itself closer to the fixed boundary condition leading to less average stress across the whole topology. The single objective solution has compliance of 0.94 N-mm which is better than any of the multi-objective solutions, as they are solved using same population size and optimization parameters as single objective.

In Fig. 11c topologies with different volume fractions are shown. For high $\eta$ solutions, the widths used are larger in size, whereas for low $\eta$ nodes are eliminated to give simpler geometries. Also, as compared to Fig. 11b, the compliance axis has more range owing to low volume fraction solutions.

### 4.2 *3-Dimensional geometries*

The applications of three-dimensional geometry are presented for cases not requiring symmetrical nodes or special nodes (as voids and non-design regions are not considered in the following case studies).

The topology optimization through python ('ToPy') (Van-Rossum and Drake 2010; Hunter 2009) is a python-based code which uses discretization of domain with multiple filters for smoothening and checkerboard correction to provide us with an optimized topology without FEA solver. It basically works on SIMP methodology and the implementation and analysis have been made fast and easy.

#### *4.2.1 Test problem 1*

The design domain is an equal sided 'cube' loaded (Fig. 12) in the center at the top with the bottom four corners fully constrained. For the solution found in literature a simple four-legged trestle is formed without any traces, citing the reason that four corners are not allowed to move at all.

### 4.3 Single objective: mean compliance

For single objective case, we take population size as 20 and maximum number of generations as 140. The fixed nodes, $k = 3$, make the total number of variables to be 43 (Fig. 13).

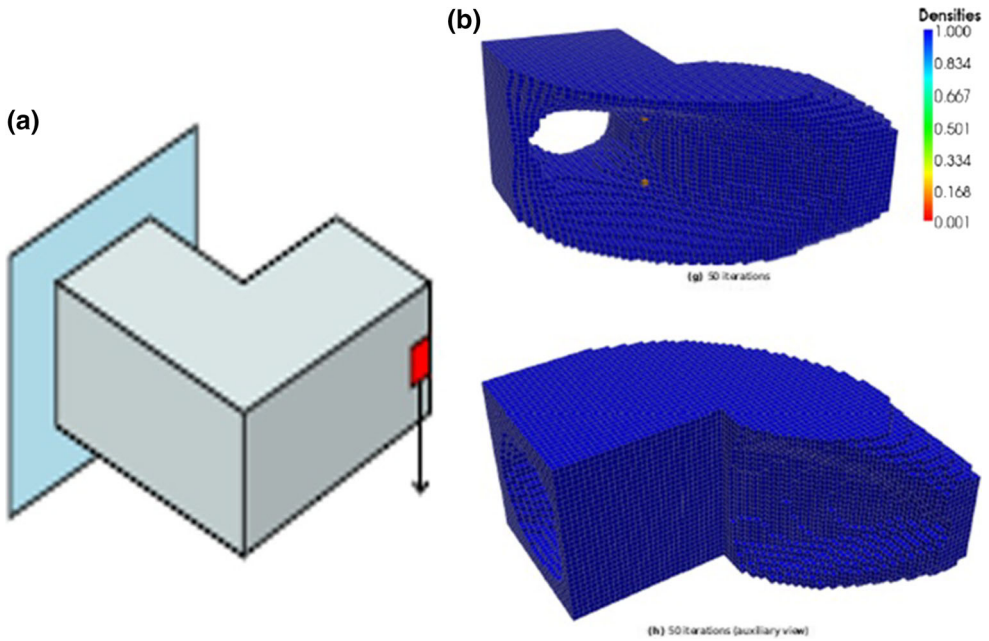



**Fig. 16** Design domain (L-shaped beam) along with literature solution (Michell 1904) comparison purposes. **a** Design domain with loads. **b** Solution

Taking the case of only 2 variable nodes ($m = 2$), the solution in Fig. 14b is reached pretty quickly, as the literature solution (Fig. 12b) demands less complexity. Also, the four leg-shaped structures are evolved, without any braces, which is the case when $m = 5$ is taken (Fig. 14a).

The introduction of more nodes demands the evolutionary algorithm to identify and eliminate the unwanted widths, for example, in the case of 5 variable nodes the numbers of width design variables are 24. Now with $p_{width} = 0.5$, widths are initialized and after the tetrahedron Delaunay triangulation, some unwanted nodes and widths have to be removed through evolution. Either they be removed or the nodes come together in a way to form the legs of the trestle. For an increased number of nodes, this would require a larger population and generations to achieve. Hence, the solution for single objective case in Fig. 14a can be improved with aforementioned optimization parameters.

### 4.4 Multiple objectives: mean compliance and volume

For multi-objective case, we take population size as 40 and maximum number of generations as 50. The total number of variables remains as 43.

Different solutions with different volume fractions are shown in Fig. 15b. For solutions having smaller topologies, it can be seen that 4 variable node connections which have been eliminated to attain leaner geometries. As the volume fraction increases, the CSG-TOM methodology tends to go either way by adding more nodes, which generates more rectangular primitives. This may result in either higher volume or increase in the widths of the already present connections. The nodes may not all lie in a plane to make a straight leg, but rather could lead to formation of braces. Again, use of more population with generations can resolve the problem of sub-optimal solution and can improve the pareto-front by

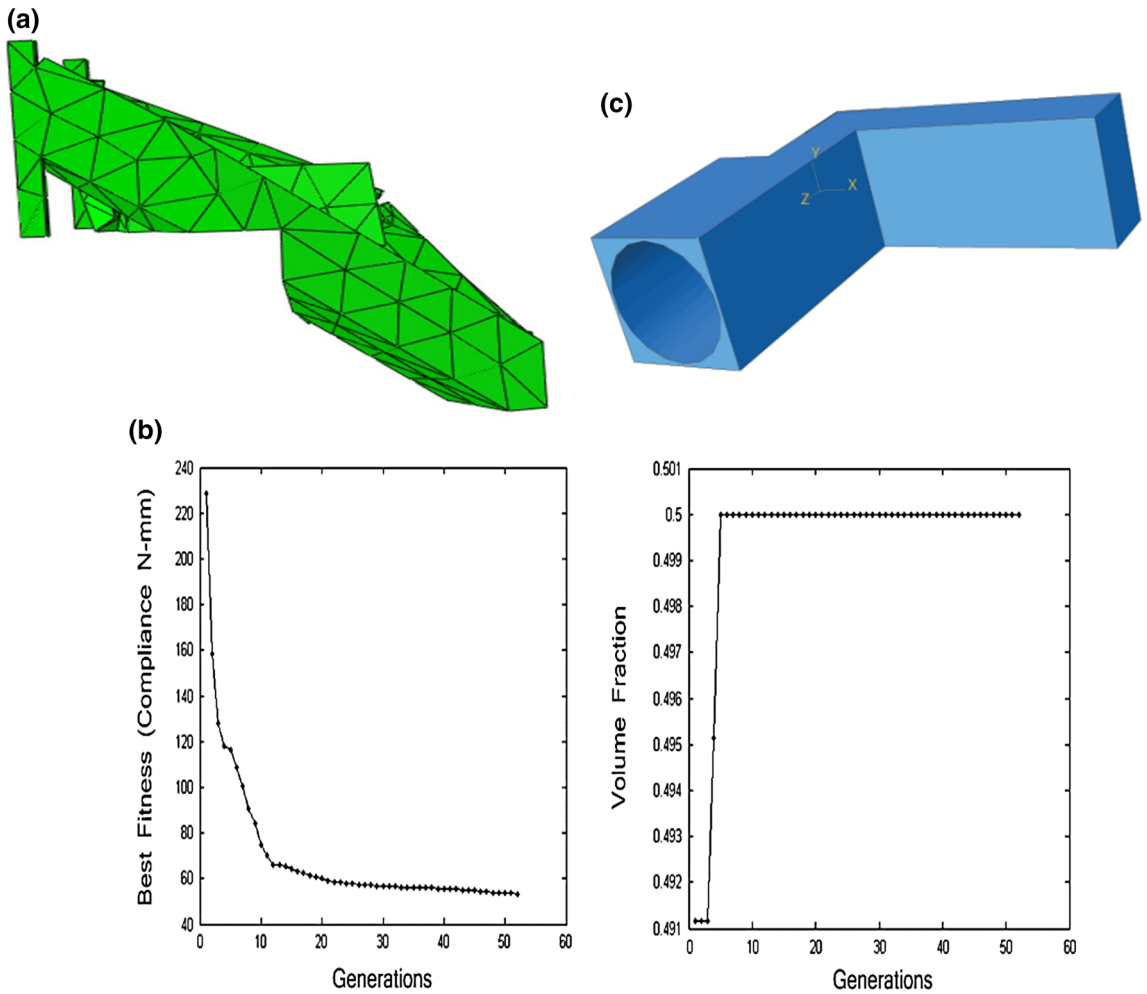



**Fig. 17** **a** Single objective solution for L-shaped beam problem, **b** graphs of mean compliance and volume fraction with increasing numbers of generations, **c** rough approximation of the literature topology for getting an estimate of the compliance value of literature solution for comparison purposes

adding more points on it, giving us diverse volume fraction topologies as well.

#### *4.4.1 Test problem 2*

The design domain is an L-shaped ('dogleg') prism fully constrained at one end as shown in Fig. 16. The domain is loaded with point load of 100 N downwards in the center on the near side of the domain at the right.

Here, we take population size as 20 and maximum number of generations as 50. The fixed nodes, $k = 3$, make the total number of variables to be 43.

The solution presented through our method cannot attain a tube-like structure through only 8 nodes with the reliance on CSG union of rectangular primitives only. But here in our solution, it can be observed that the region removed as shown in Fig. 17a is similar to that in the literature (Fig. 16b). The region around the wall is removed between the two fixed nodes. The overall shape is similar to that of the literature, when considered with the limitations. The mean compliance for the solution obtained is 52.1 N-mm. The mean compliance value for the solution in literature could not be found, hence a simplistic model as shown in Fig. 17c was created to get an approximate idea of compliance which comes out to be 50.1 N-mm.

### 4.5 Some limitations of current methodology

The limitations of current methodology lie in the basic fact that it approximates every topology through the use of only rectangular primitives. The number of nodes is a finite set and their connections are in turn limited by Delaunay triangula-

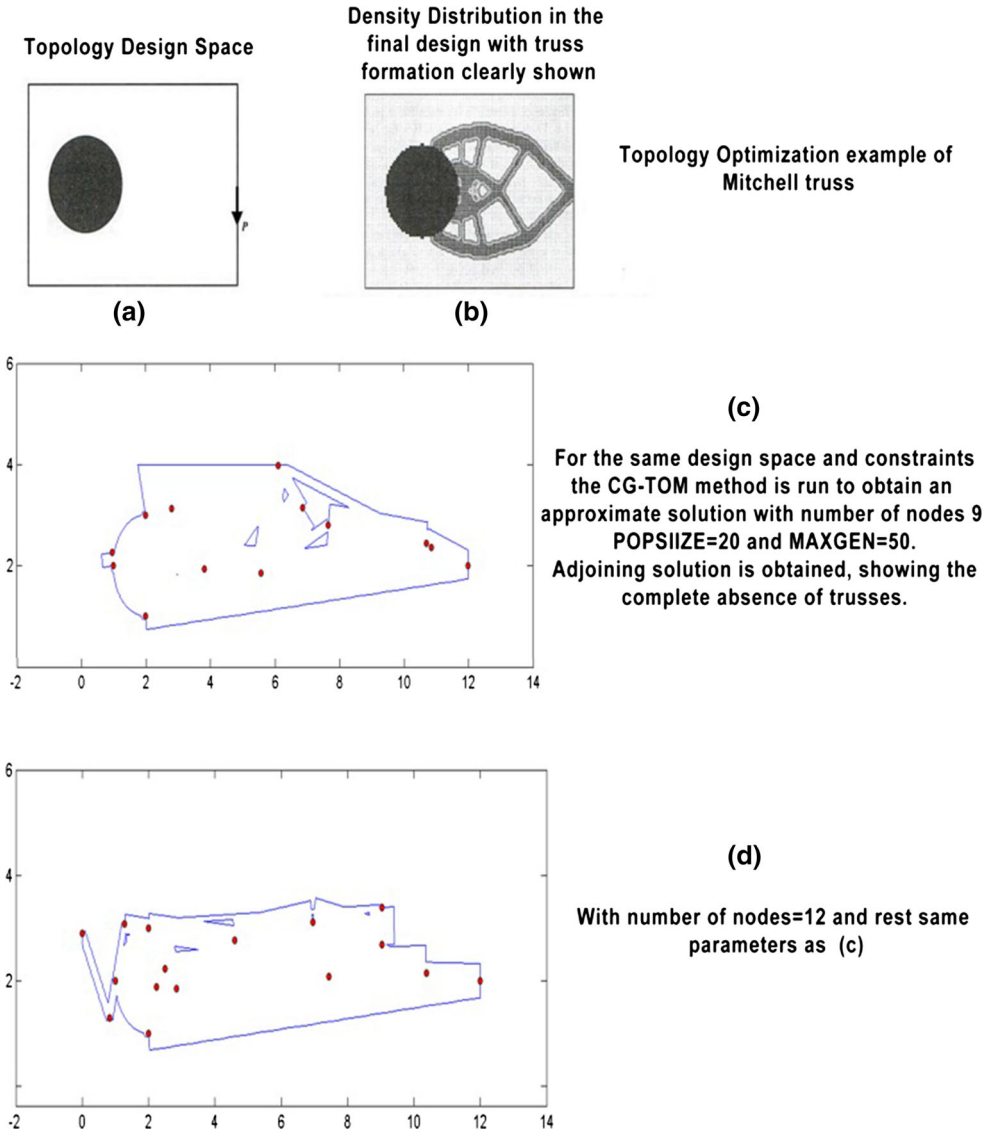
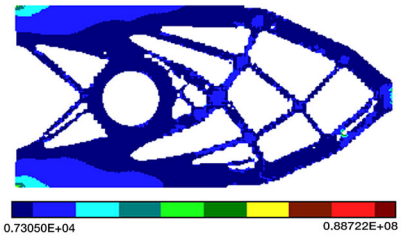
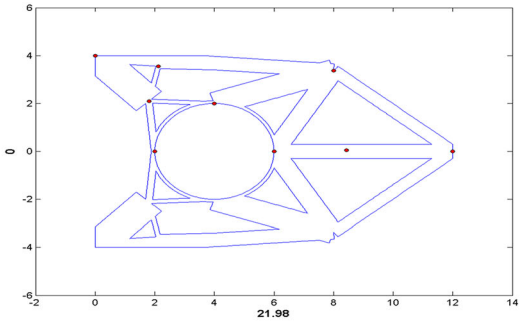



**Fig. 18** Solutions obtained for Mitchell Truss problems. The arising differences are discussed




Incorporation of Circular Hole in Topology




Solution obtained through CSG-TOM technique through implementation of symmetry condition across the domain. Initialization of points on the circle boundary has been done and fixed for all iterations.

**Fig. 19** Literature solution compared with solution obtained by CSG-TOM upon application of symmetry nodes

tion. The limited node connections might prove inadequate for proper approximations of especially curves present in the topology. The example of Mitchell trusses (Bendsoe 1995; Michell 1904) formed for the simple domain is shown in Fig. 17. As volume fraction is decreased for the Mitchell truss problem, the thicker trusses start dissociating into multiple thinner trusses, all happening between the circular non-design constraints and concentrated force node. The solution cannot be matched by that obtained through our methodology due to the complex nature of approximation involved. Although one can observe that for relatively small number of generations and function evaluations, we see that the area on the other side of the non-design constraint has been eliminated similar to solution shown in Fig. 17b. The symmetry condition without being enforced the solution seems highly irregular in shape and progresses towards the formation of local optima wherein the bulk of the material lies in between the non-design constraint and load resembling a beam-like structure.

The CSG-TOM does not have the inherent capability to identify symmetry condition or provide us with symmetric solution when design problem itself seems to offer it. If the designer identifies symmetry or wants it enforced on the domain, this information along with the number of symmetry points can be supplied to CSG-TOM, which then applies it to the domain (Fig. 18).

Other limitations include importance to certain design variables, whereas complete ignorance of some while formulation of any geometry. For example, after Delaunay triangulation, only specific widths form part of geometry and the rest do not influence the fitness value in any way. As the number of nodes is increased, the number of unused design variables increases (${}^{n}_{2}C$) nullifying the low variable advantage over SIMP (Bendsoe 1989; Nishiwaki et al. 1998) methodology. Thus, the ignored design variable widths are not guiding the optimization routine towards convergence, but rather adding to the complexity. A more focused number of design variables by identification and distinction among the whole set of possible widths need to be implemented for better optimization routine and quicker results (Fig. 19).

## 5 Conclusion and future work

CSG-TOM shows advantages in terms of population-based approach for faster pareto-optimal search in case of multiple objective, as well as enables geometric complexity control by taking number of nodes as input from user. The current work extends the new technique of CSG-TOM to include various design constraints and domain shapes in two-dimensional topologies. The primary motivation of the proposed approach is the need to develop and enhance CSG-TOM to include different test problems that could be handled by the present popular techniques of topology optimization. First, a thorough literature survey was carried out into the existing methods to try and improve the shortcomings of CSG-TOM method like reduction in computational time, slower convergence and more function evaluations. Moreover, methods for finding out suitable width limits based on node number, domain shape and size were introduced.

Thereafter, the work also extends the CSG-TOM method for three-dimensional topologies through the use of novel MATLAB–python-based software implementation with NSGA-II C code converted to MATLAB version. The compliance minimization problems with volume fraction constraints are solved and compared with those present in the state of the art. Moreover, multi-objective problems with trade-off solutions are analyzed for some problems which in turn could help decision maker to come up with better design.

The current work has built-up the basic framework required for topology optimization of any structure with complex geometry and boundary condition. The usage of these techniques could be found in smart structures, and MEMS, which have been explored by other techniques of topology optimization. The usage of CSG-TOM in the above-

mentioned areas would definitely establish it as a viable topology optimization method with practical utility, and help in popularizing the method for further research.

As a future work, the repair process can be included in the optimization procedure itself by the use of genetic programming-based topology optimization, which forms a tree structure between the nodes. Since tree structure ensures they are always connected, we always get a connected topology and final topology with no further need of repair operator.

The improvement can be done in terms of 3-D topology simplification or smoothening of rough geometries with multiple faces to a simple easy to manufacture practical geometry. Also, the establishment of primitives or methods to introduce curves (for example, through splines) (Michiel 2001; Deaton and Grandhi 2014) in the topology along with automated symmetry condition detection can be done. In addition to that, the application of many objective optimization in this field can open up new vistas to explore the trade-offs in high-dimensional requirements specifications.

**Acknowledgments** The authors thank Prof. Kalyanmoy Deb, Michigan State University and the computational facilities provided by KanGAL, IIT Kanpur. Part of the work has been jointly supported by the Department of Biotechnology, India and the Swedish Governmental Agency for Innovation Systems.

**Compliance with ethical standards**

**Conflict of interest** None.

## 6 Appendix 1

In 'Topology Repair Operation', the geometries are viewed as graph of connected nodes. A bunch of inter-connected nodes is termed as a 'Set'. Following steps are taken to obtain the final geometry:

1. Find the 'Sets' of inter-connected nodes.
2. Find the 'Special Sets' containing k fixed nodes and non-design constrained sets. If they fall in the same set, then go to Step 7.
3. Using the initial triangulation data, find the connectivity between each pair of sets which were removed by the optimization variables.
4. Obtain the graph of connectivity between sets. Weight of connection between any two sets is through the nodes joined by the minimum edge length.
5. Using breadth first search (BFS) from each set in the graph, find the minimum connections required to join all the 'Special Sets'. The absolute values of widths are taken for this new connection.
6. Join the pair of nodes for each new connection made to form a single connected 'Set'.
7. Ignore all other sets and form the final geometry using the singly connected 'Set'.

## References

Abaqus Users Manual (2006). "Abaqus"

Ahmed F (2012) Topology optimization of compliant systems using constructive solid geometry. Masters Thesis, IIT Kanpur

Ahmed, F., Bhattacharya, B., & Deb, K. (2013, January). Constructive Solid Geometry Based Topology Optimization Using Evolutionary Algorithm. In Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012) (pp. 227–238). Springer India

Bendsoe MP (1989) Optimal shape design as a material distribution problem. Structural optimization 1(4):193–202

Bendsoe MP (1995) Topology design of truss structures. Springer, Berlin

Bendsoe MP, Kikuchi N (1988) Generating optimal topologies in structural design using a homogenization method. Comput Methods Appl Mech Eng 71(2):197–224

Braid IC, Lang CA (1974) Computer-aided design of mechanical components with volume building bricks. Automatica 10(6):635–642

Cuillière JC, Francois V, Drouet JM (2013) Automatic mesh generation and transformation for topology optimization methods. Comput Aided Des 45(12):1489–1506

Datta R, Deb K (2011) Multi-objective design and analysis of robot gripper configurations using an evolutionary-classical approach. In: Proceedings of the 13th annual conference on genetic and evolutionary computation. ACM, pp 1843–1850

Deaton JD, Grandhi RV (2014) A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct Multidiscip Optim 49(1):1–38

Deb K, Agrawal RB (1995) Simulated binary crossover for continuous search space. Complex Syst 9(2):115–148

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:82–197

Deb K, Datta R (2012) Hybrid evolutionary multi-objective optimization and analysis of machining operations. Eng Optim 44(6):685–706

Guide MUS (1998) The mathworks. Inc, Natick, MA, 5

Hamza K, Saitou K (2004) Optimization of constructive solid geometry via a tree-based multi-objective genetic algorithm. In: Genetic and evolutionary computation—GECCO 2004. Springer, Berlin, pp 981–992

Hazewinkel M (ed) (2001) Bezier curve. Encyclopedia of mathematics. Springer, Berlin. ISBN: 978-1-55608-010-4

Howell LL (2001) Compliant mechanisms. Wiley-Interscience, New York

Huang X, Xie M (2010) Evolutionary topology optimization of continuum structures: methods and applications. Wiley, New York

Hunter W (2009) Topology optimization using python based open source software. https://code.google.com/p/topy/

Jakiela MJ, Chapman C, Duda J, Adewuya A, Saitou K (2000) Continuum structural topology design with genetic algorithms. Comput Methods Appl Mech Eng 186(2–4):339–356. ISSN: 0045-7825

Kudikala R, Kalyanmoy D, Bhattacharya B (2009) Multi-objective optimization of piezoelectric actuator placement for shape control of plates using genetic algorithms. J Mech Des 131(9):091007

Lee DT, Schachter BJ (1980) Two algorithms for constructing a Delaunay triangulation. Int J Comput Inf Sci 9(3):219–242

MATLAB users guide, "Mathworks, 2013". http://www.mathworks.in/help/pde/ug/decsg.html

Michell AGM (1904) LVIII. The limits of economy of material in frame-structures. Lond Edinb Dublin Philos Mag J Sci 8(47):589–597

Nishiwaki S, Frecker M, Seungjae M, Kikuchi N (1998) Topology optimization of compliant mechanisms using the homogenization method. Int J Numer Methods Eng 42(3):535–559

Querin OM, Young V, Steven GP, Xie YM (2000) Computational efficiency and validation of bi-directional evolutionary structural optimisation. Comput Methods Appl Mech Eng 189(2):559–573

Rozvany GIN, Pomezanski V, Gaspar Z, Querin OM (2005) Some pivotal issues in structural topology optimization. In: Proceedings of WCSMO, 6

Sigmund O (2007) Morphology-based black and white filters for topology optimization. Struct Multidiscip Optim 33(4–5):401–424

Sigmund O (2011) On the usefulness of non-gradient approaches in topology optimization. Struct Multidiscip Optim 43(5):589–596

Sokolowski J, Zolesio JP (1992) Introduction to shape optimization. Springer, Berlin

Tavakoli R (2014) Multimaterial topology optimization by volume constrained Allen–Cahn system and regularized projected steepest descent method. Comput Methods Appl Mech Eng 276:534–565. ISSN :0045-7825

Van Rossum G, Drake FL (2010) The python language reference. Python Software Foundation, Delaware

Wang SY, Tai K (2005) Structural topology design optimization using genetic algorithms with a bit-array representation. Comput Methods Appl Mech Eng 194(36–38):3749–3770. ISSN: 0045-7825

Wang F, Lazarov BS, Sigmund O (2011) On projection methods, convergence and robust formulations in topology optimization. Struct Multidiscip Optim 43(6):767–784

Xie YM, Steven GP (1993) A simple evolutionary procedure for structural optimization. Comput Struct 49(5):885–896

Xie YM, Steven GP (1994) Optimal design of multiple load case structures using an evolutionary procedure. Eng Comput 11(4):295–302

Xie YM, Steven GP (1996) Evolutionary structural optimization for dynamic problems. Comput Struct 58(6):1067–1073

Young V, Querin OM, Steven GP, Xie YM (1999) 3D and multiple load case bi-directional evolutionary structural optimization (BESO). Struct Optim 18(2–3):183–192