



Review article

Computational Acceleration of Topology Optimization Using Parallel Computing and Machine Learning Methods – Analysis of Research Trends

Y. Maksum^a, A. Amirli^b, A. Amangeldi^a, M. Inkarebekov^a, Y. Ding^c, A. Romagnoli^d,
S. Rustamov^{e,f}, B. Akhmetov^{a,d,*}

^a Department of Mechanical Engineering, Satbayev University, 22a Satpaev street, 050013, Almaty, Kazakhstan

^b Department of Computer Science, Saarland University, Saarland Informatics Campus, 66123, Saarbrücken, Germany

^c School of Chemical Engineering, University of Birmingham, Birmingham, B15 2TT, United Kingdom

^d School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore

^e School of Information Technologies and Engineering, ADA University, 61 Ahmadbey Aghaoglu street, Baku, 1008, Azerbaijan

^f Institute of Control Systems of ANAS, 9 Bakhtiyar Vahabzade street, Baku, 1141, Azerbaijan

ABSTRACT

Development of advanced structures using modern manufacturing methods has become attractive since they allow to improve system efficiency and performance, fuel consumption reduction, lightweighting to decrease weight and durability of structures, and many more. Designing tools such as topology optimization (TO) has contributed to such developments and facilitated in adapting new manufacturing methods such as 3D printing and computer numerical control machining in many areas of engineering and industry.

TO requires computational resources, which can be significantly complex and time consuming when complicated designs and multiphysics problems are considered. To overcome these difficulties, computational acceleration techniques have been applied together with high performance computing. In the current work, various up-to-date research studies in computational acceleration of TO methods are analysed, classified and research trends are evaluated. Thus, the results of the work clearly shows that earlier works relied on central processing unit (CPU)-based computational acceleration techniques, while latest research studies mostly consider graphics processing unit (GPU) and machine learning (ML)-based approaches. The latter got significant attention within last few years and becoming one of the research areas in computational TO. From the reviewed works, it can be concluded that in all of the acceleration techniques, solid mechanics problems were mostly studied, while a few number of research studies are dedicated to heat transfer, fluid flow and electro thermomechanical applications.

1. Introduction

Recent progress in advanced manufacturing methods [1–2] has opened up new design opportunities in many industries, including aerospace [3], robotics [4,5] and medicine [6], where multi-component structures developed via old-fashioned manufacturing processes can be replaced by a lightweight continuous structure created using cost-effective one-step additive manufacturing [7], multi-axis computer numerical control machining (i.e. subtractive manufacturing) [8] or hybrid manufacturing, which combines additive and subtractive [9]. These quickly emerging manufacturing methods have resulted in high demand for design tools, which could take the degree of freedom in designing structures to a new level. Accordingly, several commercial software such as nTopology, Gen3D, Onshape and Frustum have emerged in the market to meet the demand, while existing design and simulation software, such as ANSYS, Comsol Multiphysics, Fusion360 by Autodesk, SolidWorks and others have started to incorporate light

weighting design packages into their solution list.

Among design approaches, topology optimization (TO) is considered as the most demanded technique for developing single or multipurpose solid structures. Thus, TO deals with continuous structure, where an optimal topology is achieved by whittling away some of the material in it to create lightweight porous-like pattern within predefined internal and external boundaries and constraints. In the last decade, TO technique has been improved significantly [10] and its application towards solving complex problems has been notably increased [11]. For instance, problems such as heat transfer enhancement [12], fluid-structure interaction [13], thermo-elastic behaviour of structure [14] and improving geometry of electro thermomechanical actuators [15] were numerically discretized in 3D domains and optimized using TO techniques for better performance. Although, the computational complexity of TO is significantly increased when problems with multiple interacting physical properties (i.e. multiphysics) in higher spatial dimensions are studied. Such studies lead to substantial increase in

* Corresponding author: Bakytzhan Akhmetov, 50 Nanyang Ave, 639798, Singapore, Tel.: +65 78649 4053.

E-mail address: bakytzhan.akhmetov@ntu.edu.sg (B. Akhmetov).

<https://doi.org/10.1016/j.jii.2022.100352>

Received 28 July 2021; Received in revised form 21 January 2022; Accepted 21 April 2022

Available online 25 April 2022

2452-414X/© 2022 Elsevier Inc. All rights reserved.

computational time.

Apparently, the most practical solution to reduce the computational time of TO is to use parallel computing methods [16]. The existing methods such as – MPI (Message Passing Interface) and OpenMP, developed for central processing units (CPU) based parallel computing, as well as Compute Unified Device Architecture (CUDA) platform for parallel computing using graphical processing units (GPU) – have been incorporated into sequential TO algorithms to speed up the computational time [17]. Although, in recent years, with the rise of artificial intelligence (AI) technologies and open-source AI platforms, the application of machine learning (ML) methods in advanced manufacturing processes has been substantially increased [18]–[19]. Such trend is also notable in the field of TO, where structural topology and properties such as strength, modulus of elasticity, strain and stress fields are predicted applying trained neural networks [20].

The main purpose of the current paper is to analyse state-of-the-art works especially focused on the computational acceleration of TO, classify such works in terms of methods of computational acceleration and types of studied physics, and evaluate the research trends in such areas. Moreover, paper will try to answer to the following questions: Which numerical acceleration methods have been started to be applied frequently than others? What are the strengths and weaknesses of such methods? How technological advancements and new methods have impacted the area of computational acceleration of TO?

Thus, the paper is organized into different sections. In section 2, the classification of papers based on the acceleration technique and type of studied phenomena is given. In section 3, we discuss about computational complexity and cost of solving TO problems, as well as necessity for the reduction of computational time for large scale problems. Section 4 is considered as the main body of the paper and various acceleration approaches used for TO are discussed in it. For each acceleration approach, the research works are also provided in a tabulated form, which summarizes a type of problem studied, numerical methods, hardware, software and solvers applied for computational acceleration, including improvements in computational time or speedup. Moreover, for each acceleration method there is a brief conclusion in section 4. Lastly, general conclusive remarks are provided in Section 5.

2. Classification of research works

Up-to-date research works and papers were searched in various academic research databases, such as Sciencedirect, Springer, IEEE Xplore, JSTOR and publishers, using combinations of keywords and expressions such as, “Computational acceleration”, “Topology optimization”, “high performance computing”, “CPU-based computational acceleration”, “GPU-based computational acceleration”, “Artificial intelligence”, “Machine learning” and “Deep Learning”. Downloaded works were saved in a chronological order and further distributed into different classifications, firstly regarding their numerical acceleration techniques

followed by types of physics studied per acceleration technique. Additionally, references of each paper found through the above-mentioned approach was checked to further find past research works in similar research area. As shown in Figure 1, these papers were also classified in the same manner as the ones obtained directly from databases.

Thus, by analysing the collected works, TO studies with acceleration techniques are classified into CPU-accelerated, GPU-accelerated, ML-accelerated, respectively. The reader should understand these categories as follows:

- *CPU-accelerated*: In this category, TO algorithms numerically calculated using multi CPU cores;
- *GPU-accelerated*: Here, algorithms of TO is parallelized using multi-core graphics processors to improve the computational time;
- *ML-accelerated*: TO problems are addressed using machine learning methods, such as artificial neural networks and image processing techniques, with the purpose of reducing the computation cost.

Thus in Figure 2, up-to-date research papers reviewed in this article are shown. The total number of papers found and reviewed is 35 and it includes: (i) 11 papers in CPU-based computational acceleration, where 10 of them considered solid mechanics problems, while 1 of the works addressed both solid and fluid mechanics problems; (ii) 14 papers in GPU-based computational acceleration, consisting of 9 in solid mechanics, 1 in heat transfer, 2 have both solid mechanics and heat transfer problems and 2 in multiphysics; and (iii) 10 papers in ML-based computational acceleration, where 8 of them are in solid mechanics while other 2 in heat transfer.

3. Computational complexity of TO

The overall goal of the TO is to find an optimal distribution of the material within a design domain while allowing shape and connectivity of the boundaries in the domain to change. Optimality is achieved when objective function is minimized. Various approaches have been developed to address the optimization problem itself. As reviewed in the work [10], there are density, level set, topological derivative, phase field, evolutionary and hybrid approaches. In any type of physics problem studied using TO, whether it is heat transfer [21], solid mechanics [22], or fluid flow [23], there are several general steps to run the numerical studies as shown in Figure 3. The steps are usually started from the problem formulation, followed by finite element analysis, sensitivity analysis, filtering, updating the design variable for another iteration until convergence and obtaining the optimized structure. The computational time starts from a “run” command and finishes when the algorithm gives the final topology. Readers can refer to open access codes from works, for instance, [24] and [25]. It should be noted that the computational time does not include any postprocessing such as exporting the result to CAD software or further verification of the

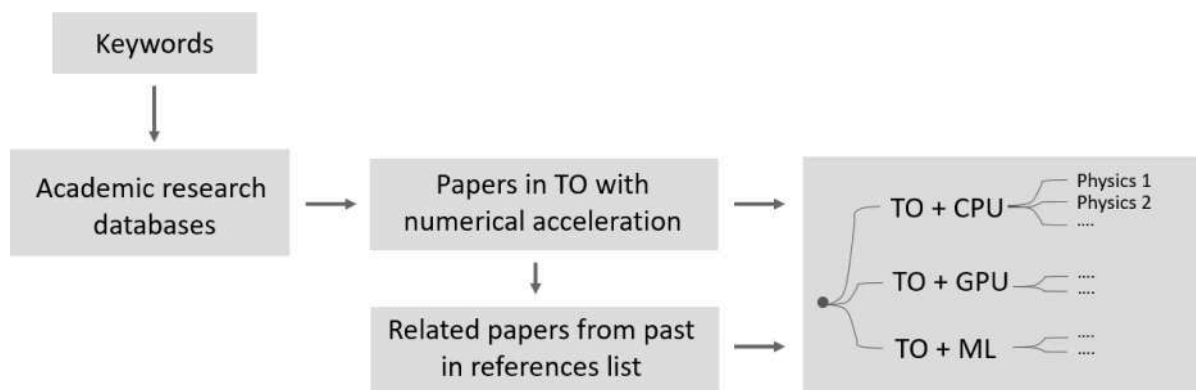


Figure 1. Finding and classifying TO studies

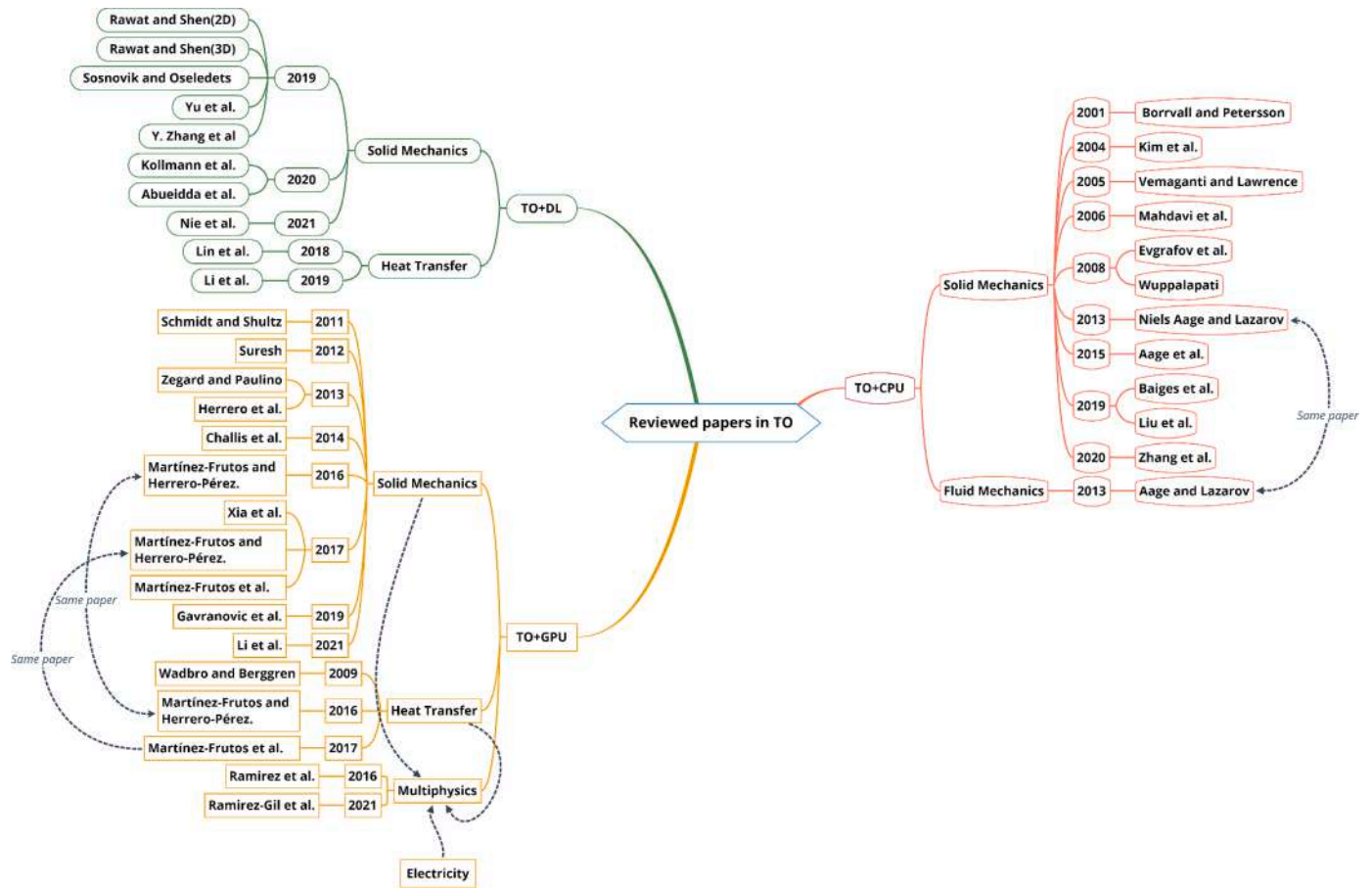


Figure 2. Classification of research works in computational acceleration of TO

obtained topology for certain applications using numerical tools. As explained in many of the works listed in Figure 2, the majority of computational time is spent to solve equilibrium equations (i.e. state equations) during FEM analysis and overall TO is achieved using iterative approach [26]. Moreover, the computational time is highly dependent on mesh size, problem dimension and design domain [27-28]. Therefore, TO requires high computational resources and time.

Following sections review acceleration techniques that have been applied for topology optimization problems up to now.

4. Computational acceleration techniques applied for TO

This section gives a discussion based on the acceleration techniques that are divided into the sub-sections. First sub-section is focused on CPU based acceleration. GPU acceleration techniques are discussed in second sub-section. And finally, DL based acceleration is discussed on the last sub-section.

4.1. CPU-accelerated TO

The application of parallel computing to science and engineering problems with the purpose of obtaining numerical results in a shorter time started several decades ago. Supercomputers equipped with dozens of CPU processors connected to each other by a communication network to distribute the computing tasks in parallel (simultaneously) were at the forefront of the field's development [29]. Borrvall and Petersson were among the first who computationally studied TO using a high performance computer [30]. They used CRAY T3E equipped with a scalable shared-memory multiprocessor, where each processor had a frequency of 300 MHz and internal memory of 128 MB. The addressed problems –

cantilever beam, crank, stool and coat hanger (Figure 4) – were optimized in terms of their topology in 3D numerical domain using the regularized intermediate density control previously proposed by the same authors [26]. The numerical domain of the particular physical problem was decomposed into subdomains (Figure 5) therefore allowing one processor to be responsible on data storage and computation with the assigned subdomain. An iterative solver, Jacobi preconditioner-based conjugate gradient (PCG) method together with the domain decomposition was used to numerically solve the equations, while optimisation part was solved using sequential convex programming. As shown in Table 1, the considered problems were parallelized and their topology obtained using various number of processors. It was concluded that 97% of the execution time was spent to solve equilibrium equations. Moreover, authors showed that 15, 18, 21 and 17 times speed up values can be achieved respectively if the problem domains are decomposed and numerically solved in parallel processors compared to sequential algorithm using a single processor. Similarly, Kim et al. [31] applied CPU-based parallel computing technique using the domain decomposition method to carry out large-scale structural eigenvalue-related design problems. To deal with the eigenvalue analysis, a parallel subspace iteration method based on PCG solver considered to be effective for decomposed subdomains was applied. There were two types of data communication were considered: between the nodes located on the boundaries of adjacent subdomains and between entire subdomains. The former is shown in Figure 5, b) and both communications was carried out using Message Passing Interface (MPI). To improve the computational time, the authors used a successive estimation scheme for eigenvalue analysis. In the optimization part of the algorithm, optimal criteria (OC) method was applied and its parallelization was explained in the work. To validate the parallel

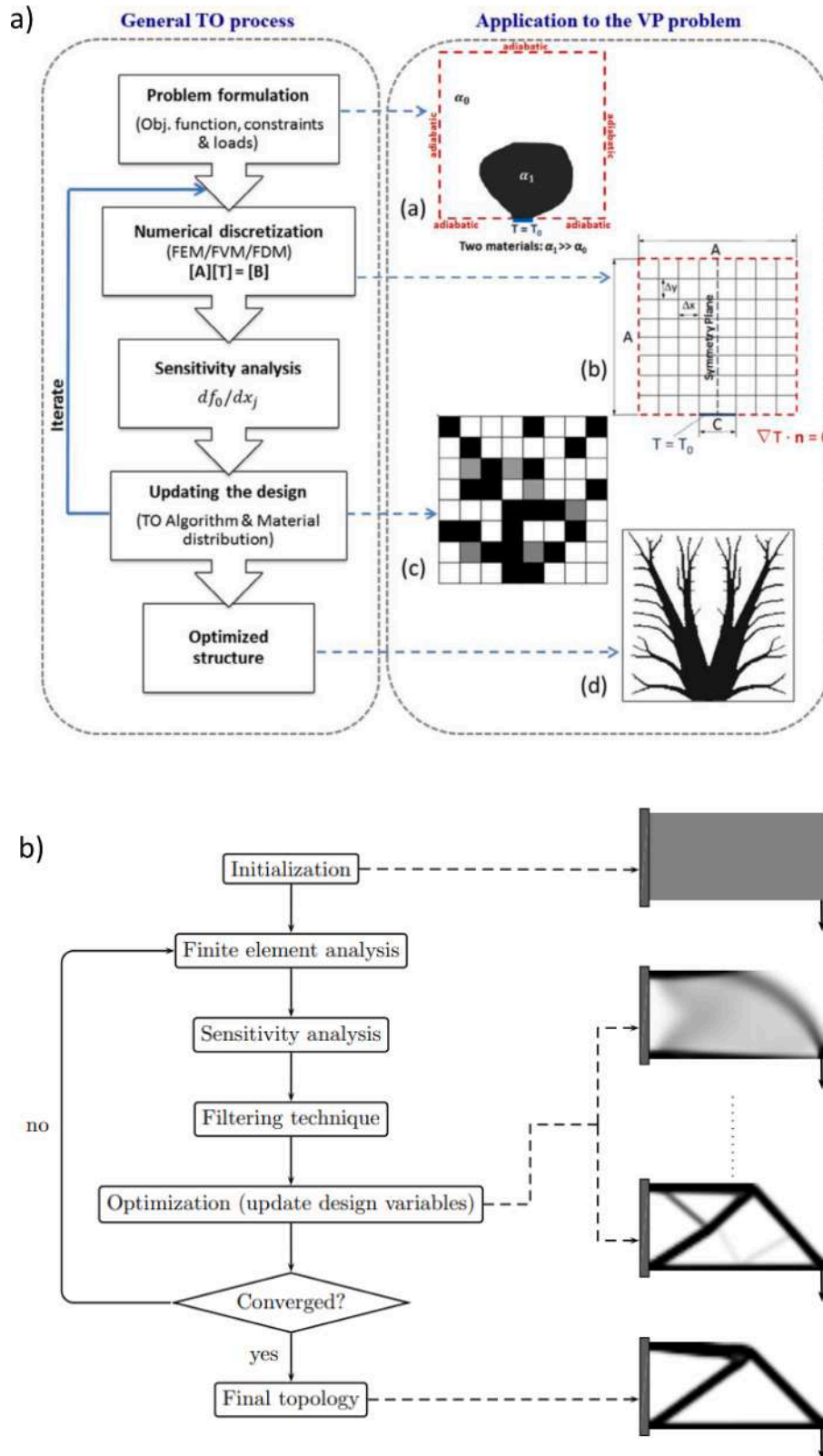


Figure 3. General steps of TO algorithm based on example physics: a) heat transfer – heat sink [21]; b) solid mechanics - Messerschmitt-Bölkow-Blohm (MBB) beam [22]

computing approach, 2D cantilever beam problem was addressed firstly, where the physical domain was divided into subdomains depending on a number of processors, which varied from 1 to 16. It was noticed that when the problem was numerically studied using a single processor,

computational time was around 3400 sec, while 16 processors could reduce the time down to nearly 600 seconds, thus, allowing to achieve about 5.6 times speedup. To further test the parallel computing approach, the authors considered elbow (Fig. 6) and optical pickup

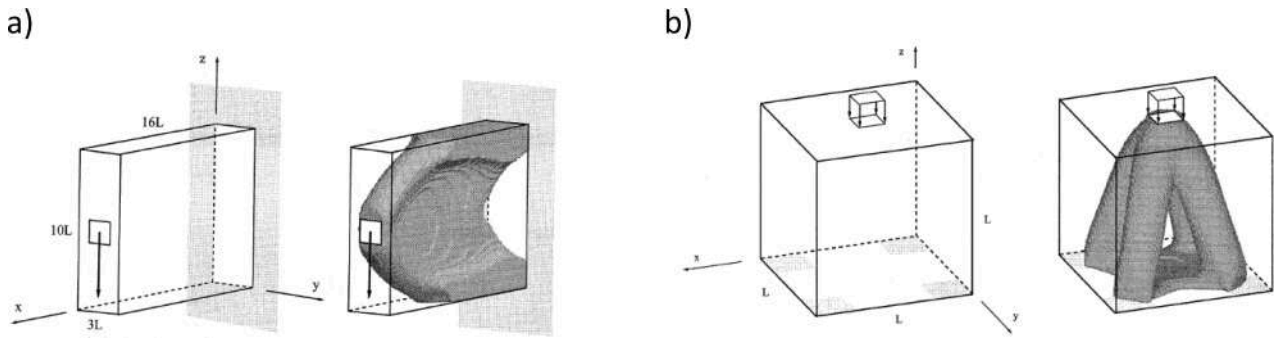


Figure 4. Illustration of numerical domain and solution of two examples: a) cantilever beam; and b) stool [30]

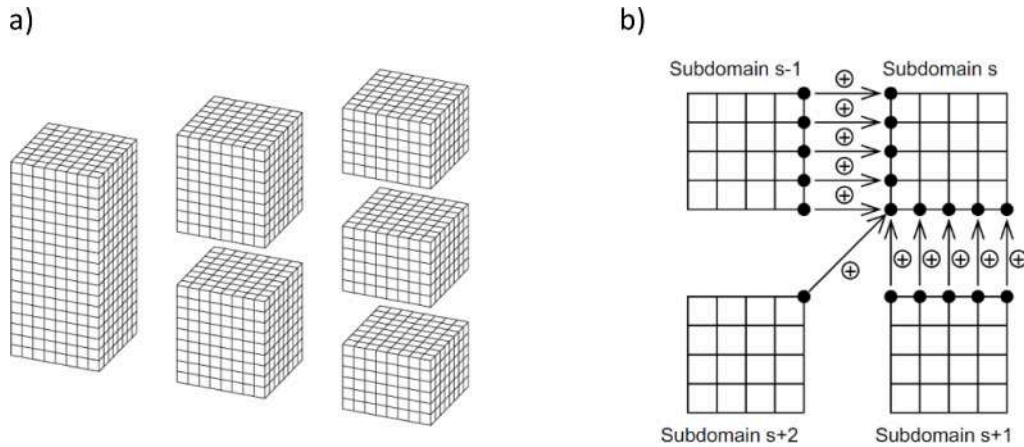


Figure 5. a) Decomposition of the original 3D computational domain into two and three subdomains as an example [30]; b) Communication that takes place at nodes on the boundaries of adjacent subdomains [31]

bobbin designs in 3D computational space using 36 and 12 CPU's respectively. The latter studies were not reported in the work neither in terms of speedup nor of parallel efficiency. In another work, SIMP and OC based TO approach was considered for parallel computing using domain decomposition method by Vemaganti and Lawrence [32]. The domain decomposition technique that the authors relied on was space filling curves, particularly Hilbert curves (Figure 7, a). In such way, non-overlapping decomposition of the domain was achieved. They tested three different solvers, which are suitable for parallelisation, namely – diagonal PCG, incomplete LU (ILU) PCG and substructuring solver that employs both static condensation and PCG – to evaluate their efficiency in reducing the poor conditioning of the stiffness matrix created by the heterogeneity of the black and white TO stiffness. Using a substructuring solver, internal nodes and interface nodes shared between neighbouring processors can be numerically solved in two distinctive steps, thus separating the calculations that need communication between processors (interface nodes) and those don't require (internal nodes). The authors proposed a multi-pass filtering technique to tackle checkerboarding and mesh dependence in the proposed parallel computing technique. The method permits to create so called “ghost” elements and use them for interprocessor communications to effectively parallelize filtering as well as mitigate mesh dependency (Figure 8, a). Object-oriented FEM tools, parallel infrastructure for finite element discretization (FIFED) and SParse Object Oriented Linear Equations Solver (SPOOLS) together with Top-Opt that contains optimality criteria and filtering schemes, and MPICH communication were applied as shown in (Figure 8, b) to solve 2D problem – a unit square with fixture and load constraints on the left and right edges respectively – using a Beowulf cluster equipped with 32 dual processor nodes. By addressing the stated simple problem (Figure 7, b), advantages of the multi-filtering technique, accuracy of the FEM solutions, and

the efficiency of the substructuring solver compared to other tested solvers were shown. For instance, it was shown that diagonal and ILU PCG solvers could not provide parallel efficiency, while substructuring solver could speedup from 320 s to 37 s when the number of CPUs carried from 2 to 32 (mesh size of 480 x 120). Mahdavi et al. [33] applied same TO approach using SIMP and OC methods. As an iterative solver, Jacobi PCG was applied due to its simplicity and robustness. It was discussed that 97% of the computation in TO is related to solve the equilibrium equations and calculation of the sensitivities. Hence, the authors proposed to apply parallel computing to them only, while heuristic filtering part of the algorithm was considered to be not worthy for parallelization. The domain decomposition method applied here was different from previous works, where the transferring of boundary data between neighbouring subdomains is not necessary, since such interactions are considered during the assembly of the internal force vector. Such approach was employed to minimize the interprocessor communications. The parallelisation of the algorithm was based on a so-called master-slave paradigm, where communications are allowed between master processors only but not for slave processors. The computational experiments were performed using Silicon Graphics Origin 2000 workstation containing 195 MHz MIPS R10000 processors. Cantilever beams with different constraints and force locations were considered as TO problems as shown in Figure 9, in 2D space and each problem was studied employing three different types of mesh sizes (Table 1). The results showed that the increase in the mesh density resulted in layouts with smoother boundary curves as expected. Furthermore, speedups of the computation with varied number of processors and mesh sizes were also provided. It was noticed that the performance plateau in all of the TO examples, started to flatten after 8 processors, indicating that no need to further increase the number of processors to 16 and 32. Thus, for the densest mesh size (160 x 80), from

Table 1

Summary of the research works on TO studies based on parallel computing approaches using CPU processors

	Research work and	Studied problem	Problem dimension	Mesh size or DOF	TO methods	Hardware	Number of processors	Parallel computing/solvers	Computational time/Speedup
1	Borrvall and Petersson, 2001 [30]	Cantilever beam	3D	Quarter domain of 24x80x128	Regularized intermediate density control	Cray T3E 300	1 to 32	Domain decomposition, MPI communication and PCG solver	3.9h/15 times (32 CPUs)
		Crank		Half domain of 40x80x80			4 to 32		11h/18 times (32 CPUs)
		Stool		Quarter domain of 96x96x96			4 to 32		21.8h/21 times (32 CPUs)
		Coat hanger		Quarter domain of 40x120x120			4 to 32		43.3h/17 times (32 CPUs)
2	Kim et al., 2004 [31]	Cantilever beam	2D	160x32	Optimality criteria method	IBM RS/6000 SP (216 Gflops)	1 to 16	Domain decomposition, MPI communication, and PCG solver with subspace iteration method	~600s/~5.6 times (16 CPUs)
		Elbow	3D	64,000 elements			36		N/A
		Optical pickup bobbin		135,807 elements			12		N/A
3	Vemaganti and Lawrence, 2005 [32]	Unit square with fixed and loaded lateral edges	2D	120 x 60 240 x 120 480 x 240	SIMP/Optimality criteria method	Beowulf cluster (1.80GHz processors, 1GB RAM at each node)	2 to 32	Domain decomposition, MPI communication and 3 types of solvers: diagonal PCG, ILU PCG and substructuring solvers	Selected substructuring solver with 480 x 240 mesh size: 320 s to 37 s (2 to 16 PCUs) speedup
4	Mahdavi et al. 2006 [33]	Cantilever beam: Left side fixed, right bottom force	2D	40 x 20 80 x 40 160 x 80	SIMP/Optimality criteria method	Silicon Graphics Origin 2000 workstation (195 MHz processors)	1 to 32	Domain decomposition, MPI communication and Jacobi PCG	~3.2 times ~2.8 times ~3.1 times
		Cantilever beam: Left side fixed, two forces applied on right		40 x 20 80 x 40 160 x 80					~2.2 times ~3 times ~4 times
		Left side fixed, domain has a hole in the middle, force-down on left bottom		40 x 20 80 x 40 160 x 80					~3.2 times ~4.5 times ~6 times
		Cantilever beam: left bottom pin-fixed and right bottom roller support		40 x 20 80 x 40 160 x 80					~3.5 times 3 times 3.2 times
5	Evgrafov et al., 2008 [27]	MBB-beam	2D	21,600 to 693,600	SIMP/Method of moving asymptotes	64-bit Linux cluster with 32 processors of AMD Opteron 846 HE (2 GHz)	2 to 96	FETI-DP with CG	2D MBB, mesh size of 1,020 x 340: Acceleration from ~1400s (2 CPU) To ~100s (96 CPU)
			3D	36,000 to 1,543,500					
		Compliant mechanism		30 x 60 x 180 75 x 75 x 150			96 8		~85 s (96 CPU) ~850 s
		Elastic surface wave-guide		160 x 160 x 64			12 to 80		~2,240 s (12 CPU) ~ 510s (80 CPU)
6	Wuppalapati, 2008 [34]	L-Bracket	2D	1250	SIMP/gradient-based optimization algorithm	LION-XO cluster with 132 nodes (2.4 GHz AMD Opteron Processors and 8 GB	1 to 12	Parallelization of Pareto curve computations	~8.3 times (12 CPU)

(continued on next page)

Table 1 (continued)

Research work and	Studied problem	Problem dimension	Mesh size or DOF	TO methods	Hardware	Number of processors	Parallel computing/ solvers	Computational time/Speedup	
					RAM)				
					LION-XM cluster with 168 compute nodes (dual 3.2 GHz Intel Xeon Processors and 4 GB RAM)			~12 times (12 CPU)	
7	Niels Aage & Lazarov, 2013 [16]	Stokes flow with minimum pressure drop and 6 outflow constraints	3D	3.35x10 ⁶ elements (state field) 7.5x10 ⁵ elements (design field)	Two-field linear elasticity formulation/ MMA	Cray XT4/XT5 18 nodes, each with 2 AMD Opteron quad cores (2.3 GHz and 16 GB memory)	144	Domain decomposition, MPI communication, direct and iterative solvers with preconditioners	~ 8 min per iteration
		Cantilever		14.7x10 ⁶ DOFs (design field) 4.8x10 ⁶ DOFs (state field)	SIMP/MMA	Cray XT4/XT5 25 nodes, each with 2 AMD Opteron quad cores (2.2 GHz and 8 GB memory)	32 to 1536		~ 5 min per iteration DOF - 25M: ~ 20 times linear speedup (800 CPU)
8	Aage et al., 2015 [35]	Cantilever	3D	480×240×240 elements	TopOpt/MMA	Cluster with 21 nodes, each consisting of two Intel Xeon 5650 6-core CPUs and 48GB memory	24 CPUs (144 cores)	PETSc/Multigrid approach/MPI communication	~ 60s per iteration (r=0.01) ~30s per iteration (r=0.1) N/A
		Roof support		512×64×86 elements			12 CPUs (72 cores)		
		Box subjected to pure torsion		82.1 million elements		cluster with 320 nodes, each with two ten Intel Xeon E5-2650 10-core CPUs	2000 cores		4.5 hours (for 2000 iterations)
9	Baiges et al., 2019 [37]	Beam-to-cantilever	2D	N/A	RTO with topological derivative concept with level set approach	LaPalma Supercomputer with 252 IBM dx360 M4 nodes, each node has 16 E5-2670 cores (2.6 GHz and 32 GB memory)	N/A	MPI communication/ HYPRE preconditioner with CG method	N/A
		Michell-type structure		Non-adaptive mesh: 36 000 elements Adaptive mesh: initial course – 2276 elements ~ 20 000 elements per processor			N/A		31 sec
							N/A		11 sec
							Strong scalability test 1: 8 to 256		~120 time (128 CPUs) 210 times (256 CPUs)
							Strong scalability test 2: 8 to 2345		~1800 times (2345 CPUs)
							Weak scalability test 3: Up to 2200		~48 times (~2200 CPUs)
							Weak scalability test: 128 to 21 120		~59 times (21 120 CPUs)
		Airplane Bearing Bracket	3D	1.2 to 120 million elements		Curie Supercomputer with 5040 Bullx B510 nodes, with 16 Intel E5-2680 cores and 32 GB memory	1536		~20 min
10	Liu et al., 2019 [28]	Cantilever beam	3D	Mesh 1: 96 × 48 × 24	Level set method with compactly	PC with Intel Core i7-6700 CPU, 4 cores and 16 GB	N/A	PETSc/MPI communication	~45 min (PC)

(continued on next page)

Table 1 (continued)

Research work and	Studied problem	Problem dimension	Mesh size or DOF	TO methods	Hardware	Number of processors	Parallel computing/solvers	Computational time/Speedup
				supported radial basis functions (CSRBFs)	memory; Work station with Intel Core i7-7820X CPU, 8 cores and 128 GB memory; Cluster (HPC1) with 5 nodes, each having two Intel Xeon E5-2670 v3 CPU, 24 cores and 128 GB memory; Cluster (HPC2) with maximum 175 nodes, each node with two Intel Xeon E5-2670 v2 CPU, 20 cores and 128 GB memory.			
			Mesh 2: $192 \times 96 \times 48$			N/A		~6.5 h (WS)
			Mesh 3: $288 \times 144 \times 72$			N/A		~2.67 h (HPC1)
			Mesh 4: $384 \times 192 \times 96$			N/A		~6 h (HPC2)
	Plate		$320 \times 320 \times 16$			N/A		~1.5 h (HPC1)
	Wheel-like structure		$120 \times 20 \times 20$			4 cores		~16 min (PC)
	Cylinder under torsional loading		$120 \times 40 \times 20$			8 cores		~1.2 h (WS)
			$288 \times 288 \times 4$			8 cores		~4.3 h (PC)
	Revolving domain with cosine curve under torsional load		$576 \times 576 \times 8$			120 cores		~2.5 h (HPC1)
			$288 \times 288 \times 4$			8 cores		~6 h (WS)
11	Zhang et al., 2020 [38]	3D	$480 \times 32 \times 120$	SIMP/MMA	Cluster -SHARCNET, each node equipped with 32 cores (2× Intel E5-2683 v4)	32 to 256 cores	PETSc/MPI communication	~5 times (256 CPUs)
	Arc bridge		$256 \times 64 \times 128$			N/A		N/A

3 up to 6 times speedups were obtained for the studied problems. Evgrafov et al. [27] criticized previous works, stating that only *parallel scalability* of TO algorithms – which characterizes the speedup of the parallel implementation for a fixed problem size depending on a number of processors – had been studied. Moreover, they mentioned that the *numerical scalability* of the solution methods, that is, the growth of their computational complexity with the problem size, were not considered in previous studies. It was discussed that the domain decomposition methods could not be numerically scalable and preconditioning methods should be universally developed in terms of their application in both 2D and 3D problems. Given these points, the authors suggested to apply a dual-primal finite element tearing and interconnecting (FETI-DP) solver, specially developed to solve large-scale finite element models on parallel computers, hoping for better computational performance results in TO problems. Studies included the application of scaling and preconditioning strategies to FETI-DP within the scope of TO. 64-bit Linux cluster with 32 nodes each having 2 GHz AMD Opteron 846 HE processor and 8 GB memory was used to run the parallel computations. As a physical problem, MBB beam was considered in both 2D and 3D dimensions using SIMP and MMA methods of TO. Although, the numerical studies showed non-promising results. When symmetric Jacobi-type preconditioning was applied, a steady increase in the number of CG iterations in FETI-DP was noticed as the heterogeneity of the subdomains increased. To further evaluate the numerical scalability

of the TO, the DOFs were changed from 21,600 to 693,600 and from 336,000 to 1,543,500 for 2D and 3D MBB beam studies. The results showed that the number of CG iterations kept increasing with higher number of DOFs (Figure 10). Furthermore, authors studied material based-domain decomposition, where solid and void parts of the domain considered separately. It was shown that such approach could demonstrate the efficiency of FETI-DP for TO problems. Asymmetric Jacobi-type preconditioning was also tested on FETI-DP but the results were not notably improved. Knowing that the augmentation matrices as an effective concept for making FETI-DP numerical scalability for 3D elastic problems, they also applied them in this study considering 2D MBB beam using 360×120 mesh size with 128 subdomains. Thus, it was shown that when material stiffness ratio was increased, the number of CG iterations was significantly increased, although, edge and weighted edge augmentation matrices were able to moderately slow down the increase. Finally, authors provided illustrative results of other studies (Figure 11) such as MBB-beam, compliant mechanism and elastic surface wave-guide in 3D domain.

Wuppalapati et al. [34] applied multicriteria decision making using Pareto surfaces on TO and as a test problem, L-bracket was considered. Two variables, weight and stiffness represented by displacement, were considered as design variables to create a Pareto curve Figure 12. The stiffness was evaluated implicitly through considering the deflection at the points of the load application. In a conventional study, two points

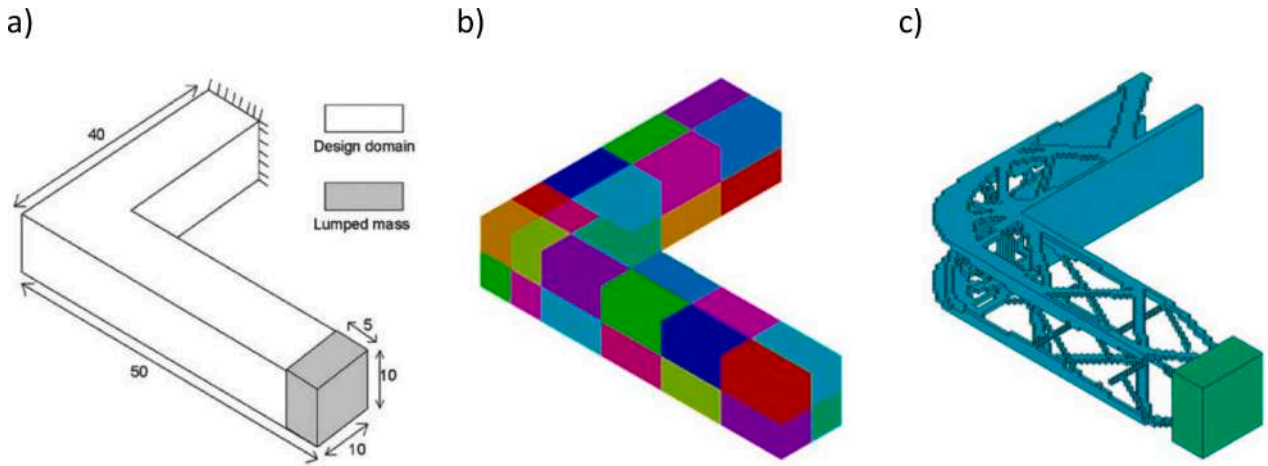


Figure 6. a) Elbow design selected for eigenvalue maximization studies; b) Domain decomposition for 36 subdomains; and c) Optimized structure [31]

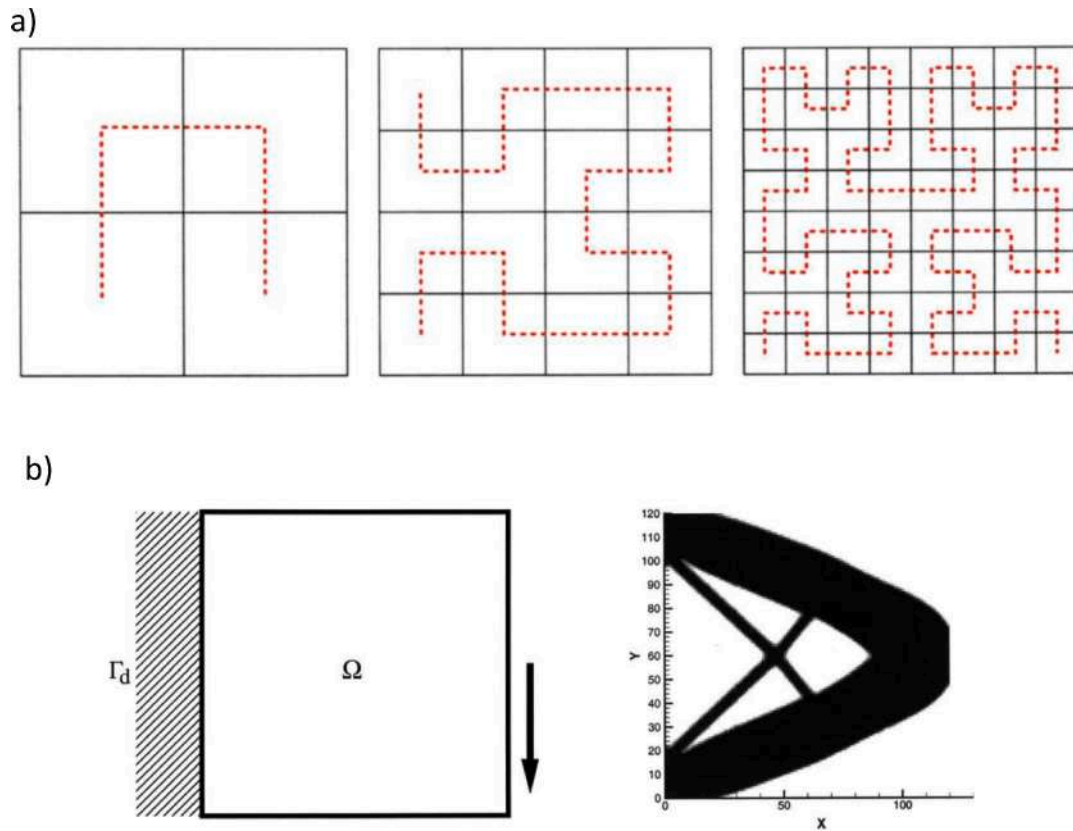


Figure 7. a) Hilbert space filling curves applied to create non-overlapping subdomains; and b) unit square design and converged topology for 120 x 120 mesh size after 1 filter pass [32]

are introduced in golden section search approach and a function or parameter is evaluated and compared in these two points. Further, a new point is introduced next to the chosen point to continue searching better design variables until the convergence. Parallel computing in such studies helps to reduce the computational time since a number of computational cores can be assigned to perform studies on several points to evaluate the Pareto curve independently. Thus, in this study, the numerical studies were parallelized using two different clusters, LION-XO and LION-XM (details are in Table 1) for faster results, where the maximum allowable deflection at the point of the applied load was varied between 1.55×10^{-3} and 5.0×10^{-3} m with equal intervals, and each topology optimization was carried out on from 1 to 12 CPUs (nodes) to

obtain corresponding values for the minimized weight of the L-bracket. In terms of FEM analysis, the domain was discretized using 1250 nodes to carry out TO using SIMP approach and gradient-based optimization algorithm. MPI-2.0 standard in Fortran 90 was used and a master-slave paradigm was applied to distribute the tasks of FEM studies among the nodes. The results show that it was possible to accelerate computational time of the Pareto curve with 24 fixed points up to around 8 and 12 times using LION-XO and LION-XM clusters. Although, in this study the application of the parallel computing for faster generating Pareto curve was straightforward and the studies did not consider any FEM level parallel computing of the considered TO studies.

Age and Lazarov [16] developed a general purpose framework that

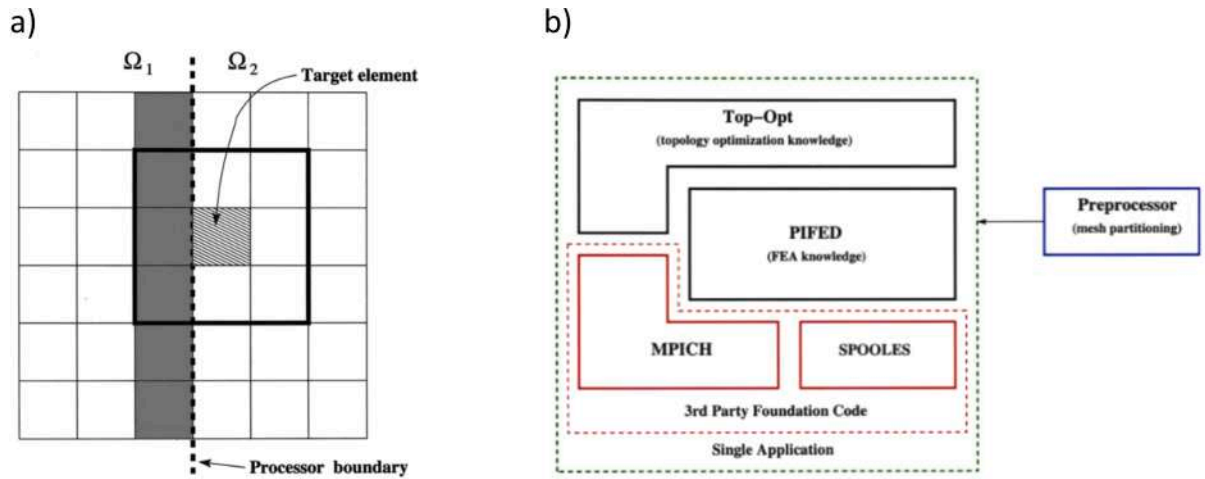


Figure 8. a) Filtering technique considered for elements residing on subdomain boundary; and b) Application architecture applied in [32]

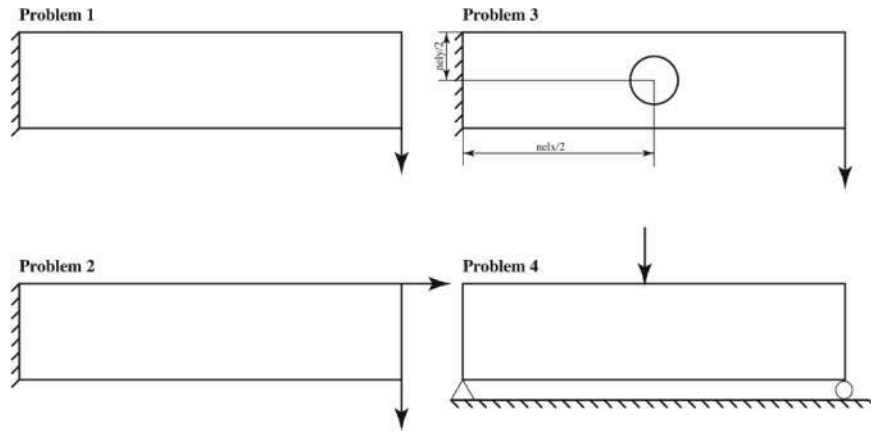


Figure 9. Illustration of selected problems for TO studies [33]

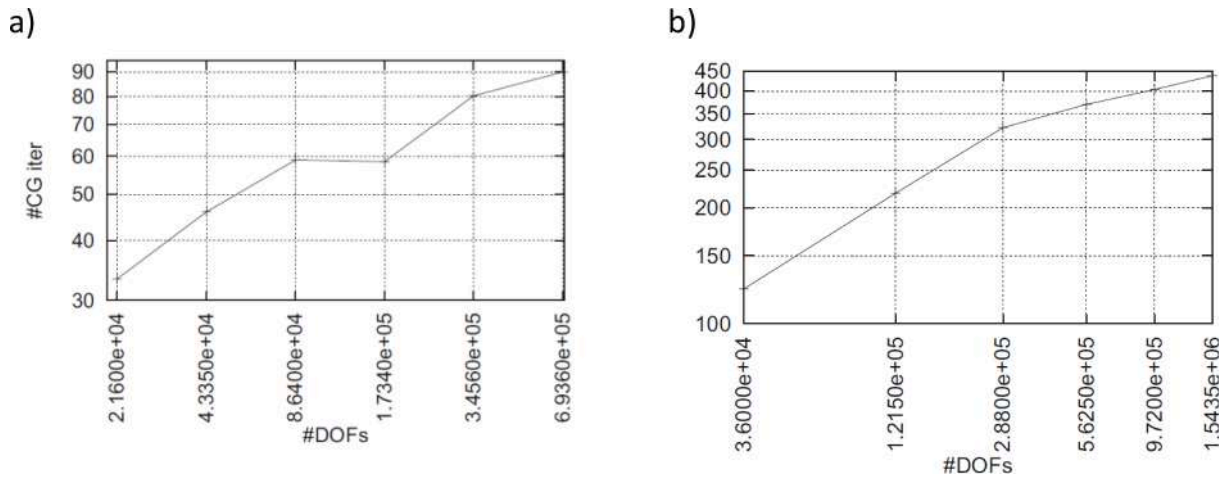


Figure 10. Numerical scalability test considering CG iteration with increasing DOFs: a) 2D MBB beam; versus b) 3D MBB beam [27]

can be applied to solve large-scale multiphysics TO problems using parallel computing approach based on an object oriented programming combined with MPI. Both direct and indirect solvers were considered in the framework relying on various freely available solvers such as UMFPACK, SuperLU and MUMPS. A graph partitioning library METIS that allows to split the computational domain into sub-domains to

minimize the communication was used, while a mesh generation was carried out by means of GMSH or CUBIT packages. The results of the computation were visualized using ParaView. As an optimizer, the MMA method was selected since it was demonstrated to be applicable for different TO problems and it is easily parallelizable. Primal/dual and dual solvers were applied for solving the MMA method. The domain

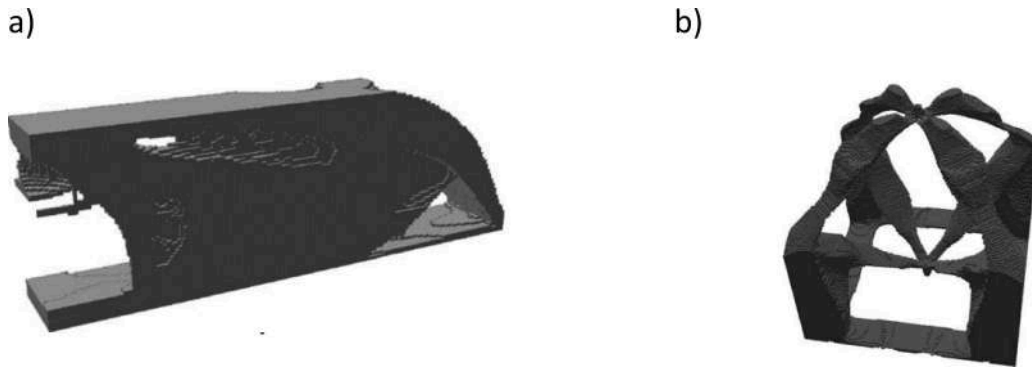


Figure 11. Optimized problems: a) MBB-beam and b) Compliant mechanism (inverter) [27]

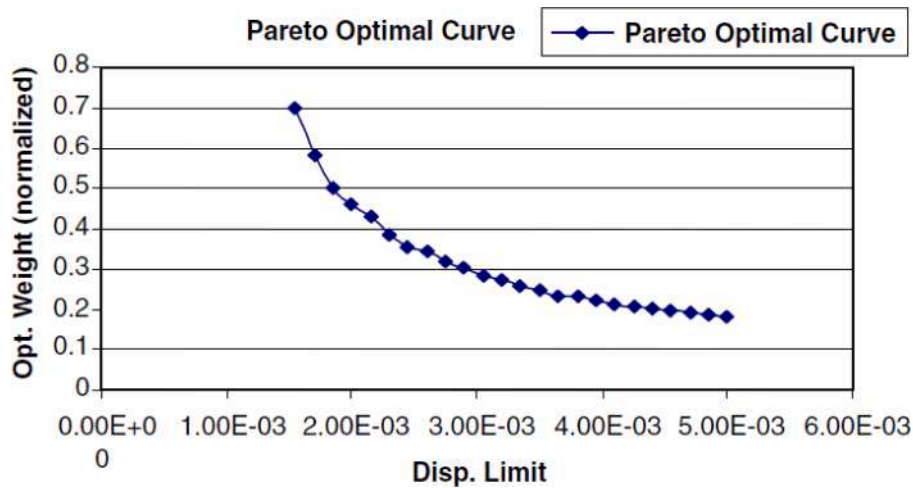


Figure 12. Pareto curve obtained for L-bracket problem [34]

decomposition technique was applied for the parallel implementation of the dual solver. To test the framework, firstly stokes flow with minimum pressure drop and six target outflow constraints was numerically studied (Figure 13). Galerkin least-squares (GLS) stabilization scheme and modified block version of the FSAI preconditioner were used to carry out the TO studies. The problem was studied using a cluster consisting of 18 nodes, where each node equipped with 2 AMD Opteron quad core with the frequency of 2.3 GHz and 16 GB memory per node. Thus, each optimization iteration took around 8 minutes. It was noticed that Karush-Kuhn-Tucker condition (which defines the optimality of the nonlinear programming) and the objective function were flattened out

after about 70 iterations while design changes reduced after nearly 400 iterations. Furthermore, the developed framework was also tested on a minimum compliance problem considering a cantilever beam in 3D domain as illustrated in Figure 14. The SIMP method was considered together with FSAI preconditioned conjugate gradient (PCG) method to carry out TO. The filtering method was solved using Jacobi PCG solver. A cluster consisting of 25 nodes, where each equipped with 2 AMD Opteron quad core CPUs with 2.2 GHz and 8 GB memory per node were used for parallel computing purposes. From the results, it was evaluated that each optimization iteration took approximately 5 minutes. The latter examples were also used to measure the parallel speed up of the

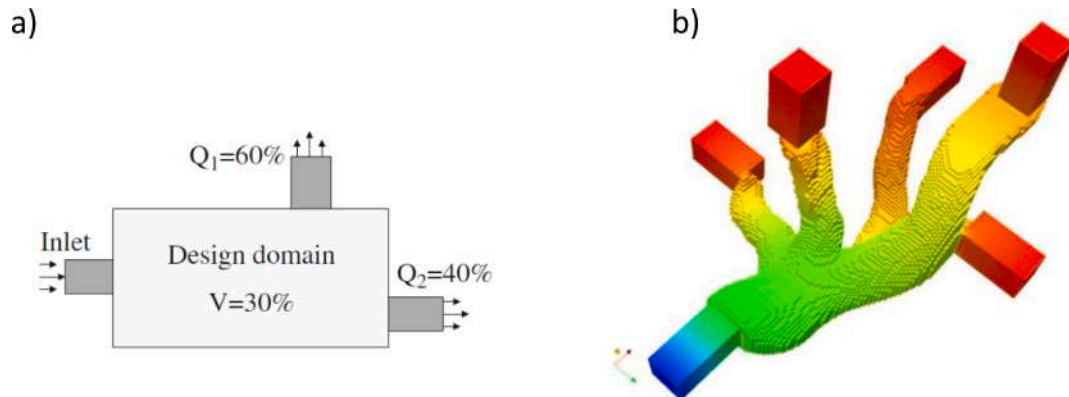


Figure 13. Stokes flow optimization: a) general illustration, where the objective is to minimize the pressure drop between inlets and outlets; b) resulting topology [16]

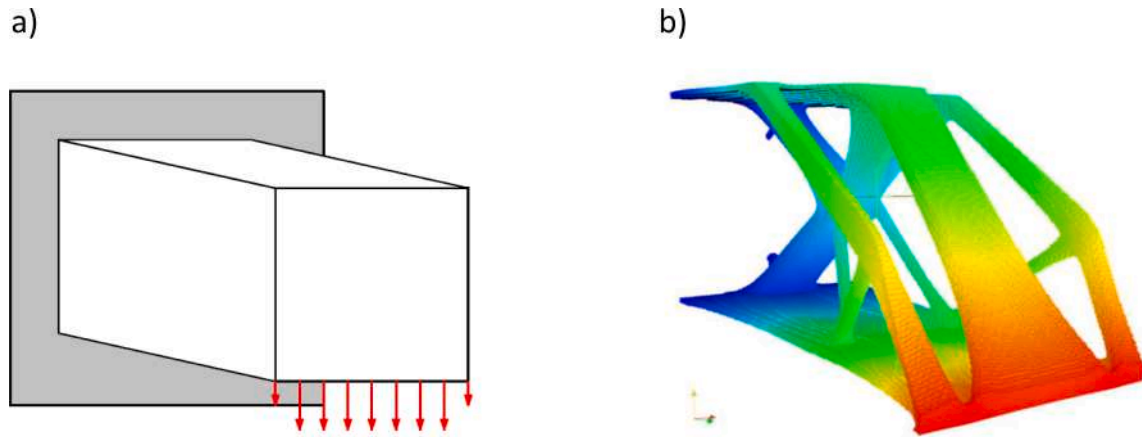


Figure 14. Minimum compliance problem: a) Design domain for the cantilever problem; b) The resulting optimized design [16]

proposed framework considering initial 10 optimization iterations. The timing of the speedup measures included all the steps from reading input data down to writing the results to the disk. According to the graphical results provided, the number of CPUs were changed from 32 up to 1536. Thus, when the mesh size set to 25M DOFs, the linear scalability could be extended up to 800 CPUs, thus achieving about 20 times speedup.

In 2015, Aage et al. [35] applied their knowledge in parallel computing of TO methods further to create an open-source framework designed to numerically solve minimum compliance problems using the Portable and Extendable Toolkit for Scientific Computing (PETSc). The information about PETSc free libraries can be found from the work of Balay et al. [36]. PETSc was chosen since it is easy to install and use, scalable to thousands of cores, portable to existing operational system such as Linux, UNIX, MAC and Windows, and users do not have carry out low-level coding due to object oriented programming approach. The authors explained at which level, users should carry out modifications on the framework to study TO problems depending on their study parameters and also showed the complexity of the components in the framework. Moreover, instructions on compiling and running of the code were provided in a step-by-step manner. It was claimed that the framework could be extended from linear to non-linear mechanics, acoustics, fluid mechanics, electromagnetics as well as multiphysics problems, although the authors limited their examples in the paper to minimum compliance studies. The selected examples were run using a computer cluster with 21 nodes, each consisting of two Intel Xeon 5650 6-core CPUs and 48GB memory. As a first example, the cantilever problem, which was studied in the earlier work [16], is considered here again. Thus, using $80 \times 240 \times 240$ elements, the minimum compliance problem was solved for the filter radii $r_{min} = 0.01$ to 0.1 while allowing

only 12% material in the domain. For the smallest and largest filter sizes, the achieved average iteration time was 60s and 30s respectively. A design of roof support especially for the Qatar convention center using the framework optimized with the target of 12% final volume fraction using 12 CPUs (72 cores). The selected filter radius was $r_{min} = 3.0$ and domain was divided into 3.1 million elements ($512 \times 64 \times 86$). As a third example, $1 \times 1.2 \times 1.2$ box subjected to a pure torsion was considered, but it was numerically studied using a bigger cluster with 320 nodes, where each node consisted of Intel Xeon E5-2650 10-core CPUs. Using 82.1 million elements and 2000 cores, the total computational time for 2000 iterations lasted approximately 4.5 hours to achieve small volume fractions (1% and 0.5 %) and the resulting topology is shown in Figure 15. Other examples were studied by changing the physics from static linear elasticity to elastic material design. Thus, an isotropic 3D microstructure with Poisson's ratio of -0.80 was optimized for volume fraction of 0.08 using 8 million elements. PDE-type sensitivity filter was applied with the initial radius of 10%, which further decreased to 2.5%. Lastly, the framework was tested on the maximization of the bulk modulus with the target volume fraction of 30%. The number of elements for the numerical studies and CPUs were 23.8 million and 40 respectively, and it took about 60 seconds to run an iteration, which includes filtering and design update.

Baiges et al. [37] addressed the computational challenge relating to topology optimization under uncertainty (TOUU) of continuum structures, which mainly comes from finite element methods applied on complex geometries and applied uncertainty propagation methods that need statistical metrics evaluation. Among TOUU methods, robust topology optimization (RTO) approach that deals with uncertainties in loading conditions, was chosen for the study. The authors considered a

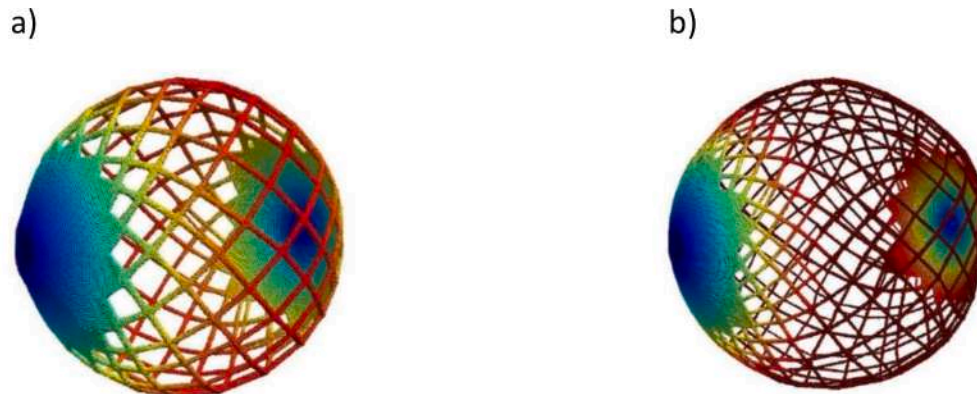


Figure 15. The results of the TO regarding $1 \times 1.2 \times 1.2$ box under pure torsion: a) 1% volume fraction and filter radius $r = 0.003$; b) 0.5% volume fraction and filter radius $r = 0.001$ [35]

two-level parallel processing strategy that permits to carry out efficient parallel computation using distributed memory system, where stochastic nodes are distributed over the memory network. The parallel computing in each stochastic mode is further conducted through domain decomposition method and adaptive mesh refinement technique in each subdomain as shown in Figure 16. MPI was deployed for parallel communication and to improve the idle time of CPUs for better parallel performance, the load balancing strategy was used to level out the workload among subdomains (i.e. corresponding CPUs). RTO was based on topological derivative concept combined with level-set approach. As a hardware, LaPalma Supercomputer consisting of 252 IBM dx360 M4 computing nodes, where each node has 16 cores with 2.6 GHz and 32 GB of RAM memory, was used to carry out the parallel computing test studies. Selected problems were beam-to-cantilever, Michell-type structure design and airplane bearing bracket. The former two was studied in 2D, while the latter was optimized in 3D domain. Thus, the purpose of the beam-to-cantilever studies was to show how level of uncertainty influences on the shape of the optimal topology as shown in Figure 17. The studies carried on the Michell-type structure was mainly for evaluation of adaptive and non-adaptive mesh performance. 36 000 elements were used in the case of the non-adaptive mesh, and it took 141 iterations with the total time of 31 seconds to achieve the final result. On the other hand, adaptive mesh-based simulation, which started with 2276 elements of course mesh, with further refinements allowed after 50 and 100 iterations, took only 11 seconds with corresponding 128 iterations. Moreover, the results showed that the adaptive mesh-based studies could provide better solution, where compliance was about 3% lower than the uniform mesh. Different types of scalability tests were performed using Michell-type structure using various types of supercomputers and number of cores which can be found in Table 1. The studies on TO of airplane bearing bracket were conducted mostly on various target volumes, starting from 0.5 to 0.1, including finite element size which varied from 1.2 million up to 120 million using 1536 processors. One of the optimized result are shown in Figure 18.

Liu et al. [28] developed a fully parallel topology optimization method designed for minimum compliance problem using parametrized level set technique with compactly supported radial basis functions (CSRBFs). Here the full parallelization refers to the computational solvability of the whole algorithm in parallel manner. The computation

of the conventional level set method relies on the solution of the Hamilton-Jacobi equation. Although, there are several difficulties that limit the efficiency of the method. To avoid them, the authors suggested to parametrize the conventional level set function using set of CSRBFs. The latter allowed to transform Hamilton-Jacobi equation from partial to ordinary differential equations which could be solved using a first-order forward Euler's method. The domain decomposition method was applied to carry out parallel computing together with MPI communication. PETSc library was selected for solving the liner system of equations, where, in particular, Generalized Minimal Residual method together with the Jacobi and multigrid preconditioners were applied to solve the equations. Example studies were run using four different computing systems as listed in Table 1. A cantilever beam optimization was conducted in all computer systems, using different types of mesh. Furthermore, a plate type and wheel-like structure optimizations were discussed. The results of the wheel-like structure optimization using non-uniform mesh are illustrated in Figure 19. Similar optimization studies were conducted on a cylinder (Figure 20) and revolving geometry and corresponding mesh size, computing systems and computational times are listed in Table 1. Furthermore, the authors showed the percentage share of computational time regarding solving the state equations of FEA, obtaining unknown variables and time spent for pre-processing, sensitivity analysis, updating level set function and other steps in the algorithm. Comparative analysis showed that more than 80% of the computational time is spent on solving FEA equations.

Zhang et al. [38] proposed a parallel computing framework to solve density-based TO using inherent strain method (ISM) especially for laser powder-bed fusion (LPBF) additive manufacturing. With the help of the framework, a support structure of LPBF was optimized to reduce its volume and printed parts' deflections. ISM was employed for the prompt prediction of residual stress and distortion, where it was calculated by means of incompatible strain as well as thermal and plastic strain initiated by nonuniform temperature distribution during the manufacturing process. Thus, TO studies mainly divided into two: TO with gravity only (GraOnly) and TO with gravity and residual stresses (GraAndISM) calculated from ISM method. The framework developed by Aage et al. [35] was considered and it was combined with the parallel computing module of ISM developed by the authors. Mesh sensitivity studies were performed firstly, and $480 \times 32 \times 120$ mesh size was

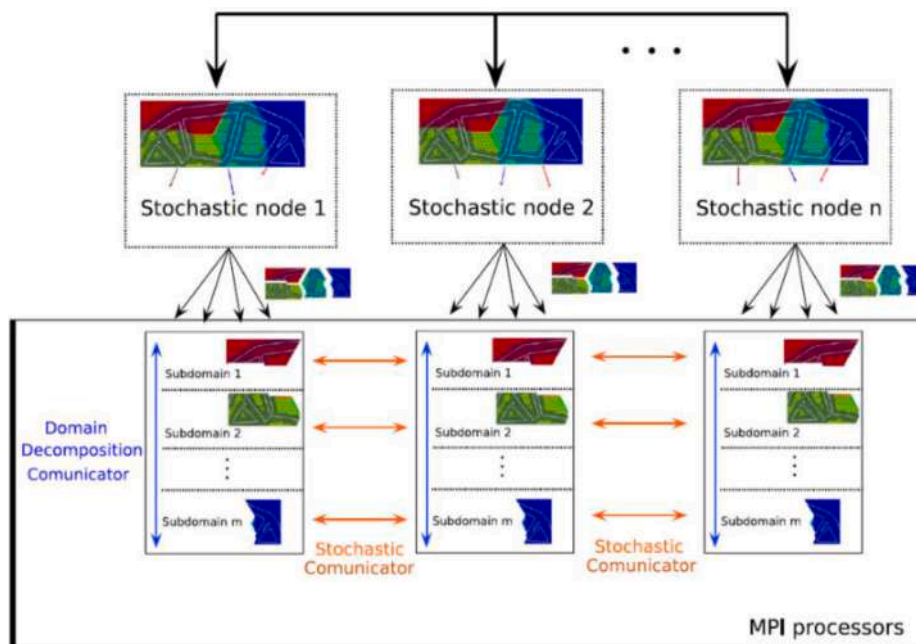


Figure 16. Two level parallel processing technique [37]

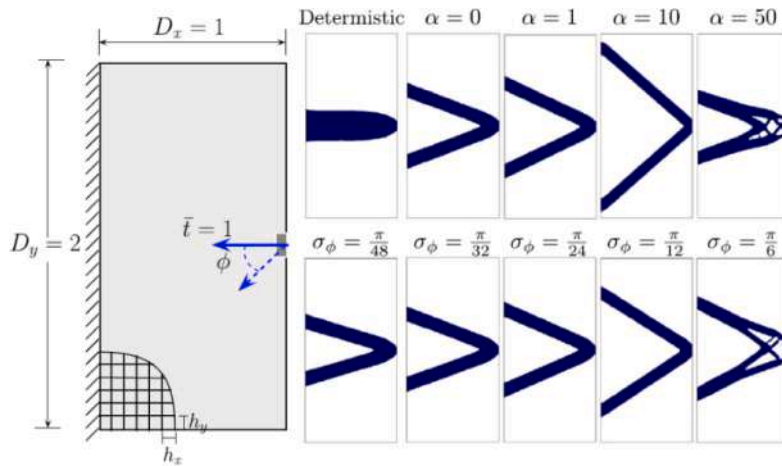


Figure 17. Illustration of the design domain and boundary conditions, which also includes deterministic design for different values of α and levels of uncertainty in the unit load σ_ϕ [37]

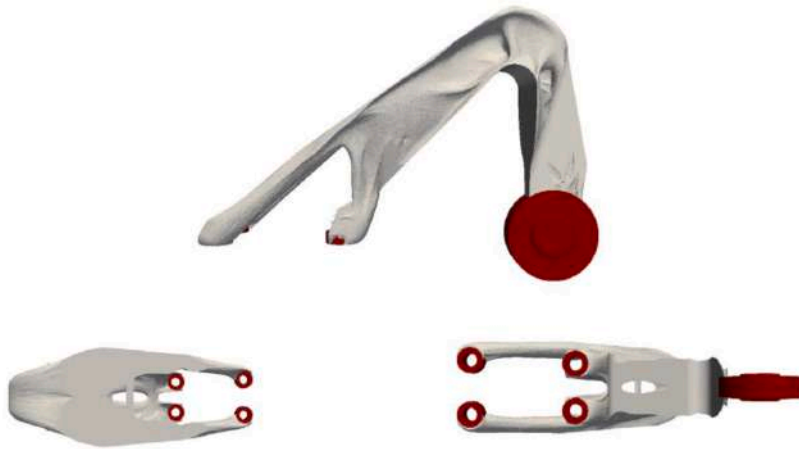


Figure 18. TO of airplane bearing bracket for target volume of 0.2, illustrated with side, top and bottom views [28]

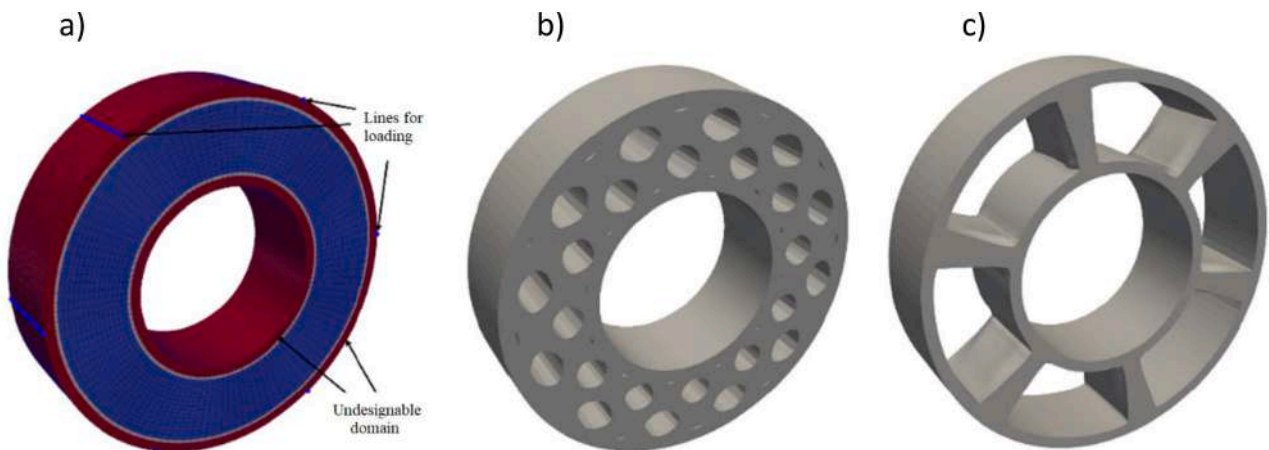


Figure 19. TO of wheel-like structure: a) Non-uniform mesh illustration; b) Initial distribution of material domain; c) Final optimized structure [28]

selected as the result of the study. The design domain as shown in Figure 21, a) was optimized under above-mentioned two TO studies, where the results are illustrated for various target volume fractions - 0.3 to 0.65. The maximum deflections of the structure supported with various topology optimized structures were further experimentally and numerically studied. The attention was also given to the material cost

reduction by means of support structure design improved with TO approach. To evaluate the speed up, authors studied GraAndISM0.3 using the parallel computing SHARCNET with up to 8 nodes. Each node had 32 cores, therefore, the number of cores varied for parallel computing was from 32 to 256. The results show that nearly 5 times speedup. Furthermore, an arc bridge geometry was further studied using

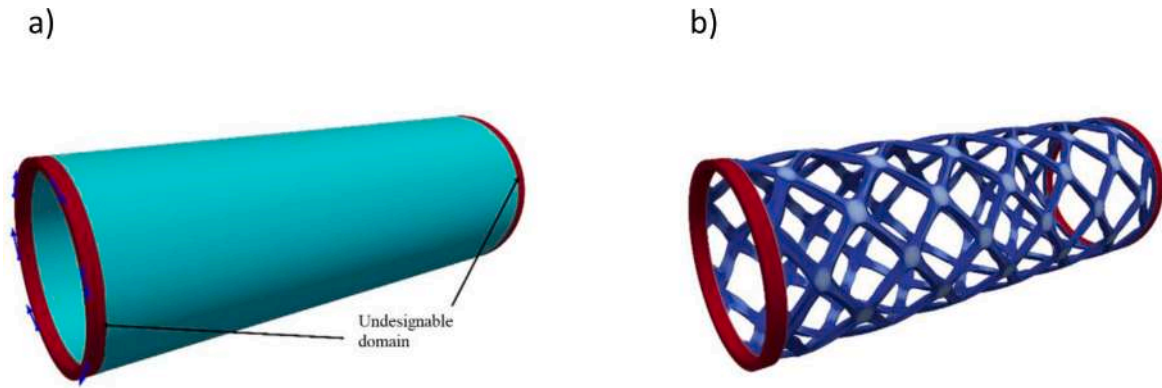


Figure 20. Illustration of the cylindrical design TO: a) computational domain and pure rotational boundary conditions; and b) obtained topology [28]

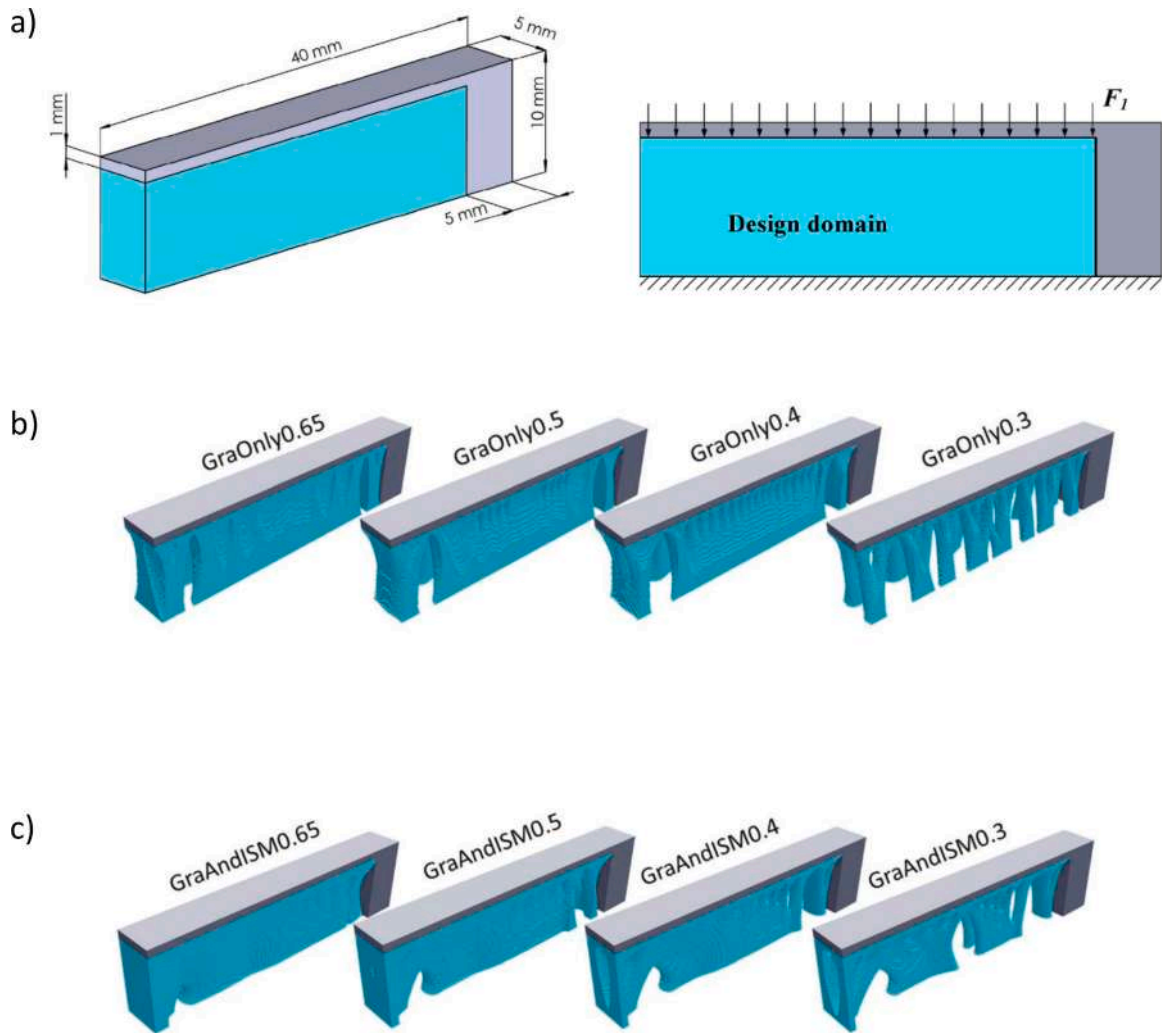


Figure 21. a) Illustration of the cantilever beam (grey) with support structure (blue); b) GraOnly study results; and c) GraAndISM study results [38]

the mesh size of $256 \times 64 \times 128$, for the same formulations: GraOnly and GraAndISM. The results were discussed based on the deflection studies, but parallel computing approach was not considered for the latter geometry.

As can be noticed from the chronologically reviewed works in CPU-based acceleration of TO, earlier studies considered how algorithms of TO, mostly SIMP method based approach, could be effectively parallelized among many CPU cores using domain decomposition and various

existing solvers, which parts of algorithms are computationally intensive and worth for parallelism. In the majority of the works, solid mechanics problems were considered with various TO variables such as void fraction, filter density, location and magnitude of applied forces as example problems of their research. Various supercomputers and clusters were deployed to study these problems using their multicore capacity, where earlier studies used up to 32 cores, while some of the later works used more than 100 cores. It was clear that with the rise of computational

capacity in hardware, a number of CPUs applied in parallel to study TO problems were more accessible since research centres started to install high capacity computers. The latest works in this area included development of software like frameworks especially designed for TO studies, where users do not have to know every line of the algorithm but still can conduct TO. Some of the latest studies included specific geometries and 3D printing applications, including TO under uncertainty and level set methods.

4.2. GPU-accelerated TO

Recently, the reduction in computational costs has been achieved by increasing the level of parallelism. One of the integral parts of modern high-performance systems is the use of graphics cards for scientific computing. The difference between CPU and GPU is in the design approach. CPU is a multi-core processor with many instructions, and GPU is a many-core processor with a smaller set of instructions. This allows the GPU to handle a large number of concurrent threads. Since TO is an iterative technique, meaning that the current iteration depends on the previous iteration, it eliminates the possibility of parallelizing the whole process. That's why GPU is mostly used as a co-processor. Additionally, in order to show the effectiveness of the GPU acceleration technique, authors usually compare it with the CPU accelerated techniques. Thus, some of the papers reviewed in this section may have both CPU and GPU acceleration techniques with the comparison.

Several investigations of GPU accelerated TO problems were conducted in the past years. Wadbro and Berggren [39] showed the use of GPUs to solve large-scale pixel-based material distribution problems. Authors considered the problem of distributing high and low conductivity inside a rectangular plate to obtain even temperature distribution. The rectangular 2D plate as shown in Figure 22 was insulated along boundary Γ_N and had a constant temperature at Γ_D . It was heated by sources uniformly distributed within the domain. By assuming that plate was made of solid material with heat conduction properties that are inhomogeneous (spatially varying) but isotropic (equal in each direction) and with a limited amount of high conductivity material ($k = 1$), and unlimited amount of low conductivity material ($k = 0.001$), the aim was to distribute these two materials, so that the temperature field will be as even as possible. Numerical study was performed on a regular computer equipped with an Intel Core 2 Duo running at 1.86 GHz and an NVIDIA 8800 GTX based graphics card. Figure 23 shows the results for the iteration time at different problem sizes, where the solid line represents the computational time on GPU, dotted line is the time on a single core and dashed line on HPC cluster. It was observed that GPU-based solver starts to perform faster than CPU-based one as the number of element sizes increases.

Schmidt and Shultz [40] investigated topology optimization based

on the SIMP approach. Authors focused on using the GPU as the complete state equation solver, rather than coprocessor. This was achieved by solving the state equation matrix-free. The conjugate gradient (CG) method was used to solve the state equation. Overall, a 3D optimization procedure was very similar to the studies in [24]. Both CPU and GPU implementations were done and compared. CPU implementation was on AMD 6176 SE with 48 cores, while GPU studies were carried out using NVIDIA Tesla M2050. TO was tested on a 100x100x200 mesh for cantilever beam problem. Figure 24 shows the time needed for sub-routines such as CG iteration, computation of matrix-vector product, residuum and copying. It can be seen that GPU implementation outperforms the CPU implementation. Additionally, it was observed that the time needed to compute matrix-vector product was dominated for the CPU implementation, and inner products and norms for CG method were dominated for the GPU implementation.

An algorithm and implementation for large-scale 3D topology optimization were presented by Suresh [41]. Authors chose matrix-free approach of the Jacobi preconditioned conjugate-gradient (Jacobi – PCG) method for solving a linear system of equations. Studies were based on 2D Pareto topology optimization (PareTO) method that was presented in authors' previous work [42]. Numerical experiments were conducted on Intel I7 960 quad-core CPU and NVIDIA GeForce GTX 480 GPU with 480 cores. Several problems were tested. Table 2 below demonstrates results for total number of FEA iterations, compliance ratio, CPU and GPU time. In another research, Zegard and Paulino [43] investigated the feasibility of finite element methods and topology optimization for unstructured meshes on GPUs. They aimed to implement an algorithm with acceptable results. TO loop was divided into six modules: FEM assembly, boundary conditions, solver, sensitivity, filter and density update. CPU and GPU versions were written for each of these modules. Four different approaches were considered: all-in CPU TO loop, all-in GPU TO loop and combination of both, as shown in Figure 25. Cholesky based solver was selected for the GPU, while LAPACK [44], linear algebra package distributed in AMD Core Math Library was used for the CPU solver.

Three types of 2D physical problems: topology optimization of bike frame, MBB and MBB with hole were considered. Table 3 below shows the runtime of these three problems for different approaches. It can be seen that the fastest approach is to use the hybrid GPU-X. Additionally, it was observed that solver takes 90-95% of the total runtime.

Herrero et al. [45] presented a GPU implementation of the level set method for TO. The approaches based on level set method consider the boundary of the material as a design variable. Hamilton Jacobi equation is then used to represent the evolution of the level set function. The flowchart of TO using level set is shown in Figure 26. Implementation of the evolution of level set function happens in the third stage (updated design). On the fourth stage, the padding (to represent void elements at

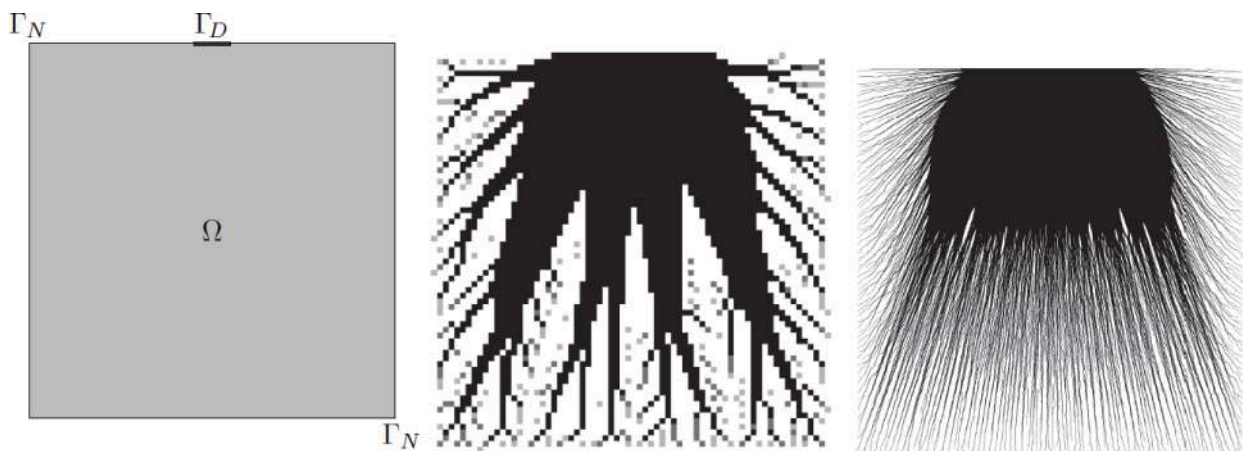


Figure 22. Computational domain (left) and results for $N=70^2$ (middle), $N=1050^2$ (right) elements [39]

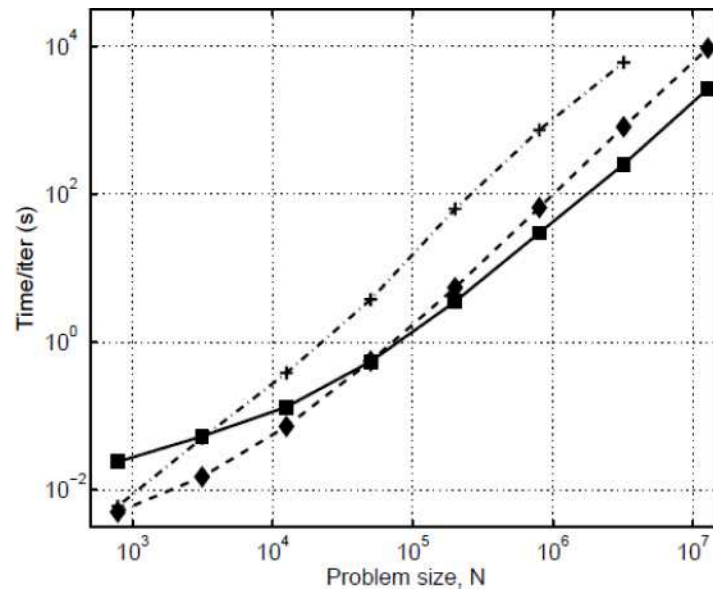


Figure 23. Iteration time for different problem size [39]

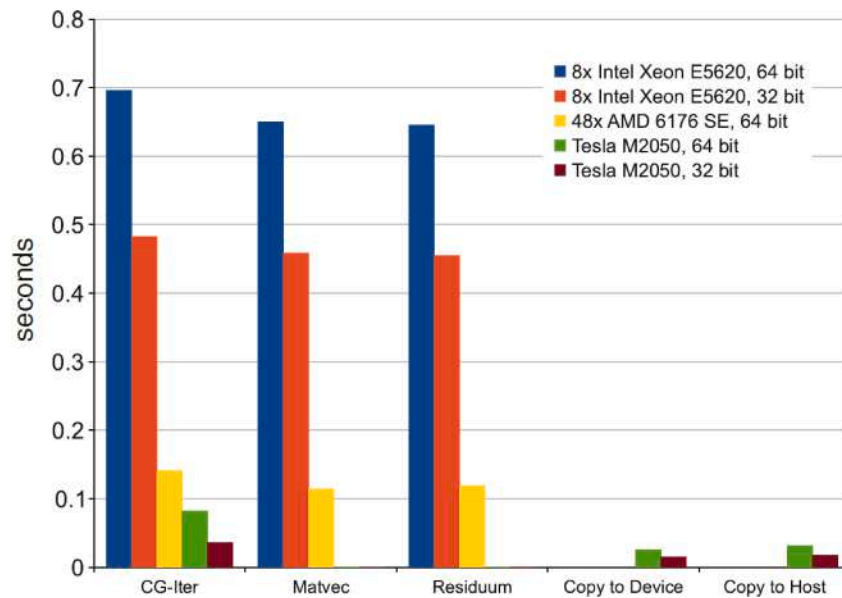


Figure 24. CPU and GPU time needed for different subroutines [40]

Table 2

Summary of computing times and compliances

Name of part & volume fraction	DoF	PareTO #FEA	$\frac{J}{J_0}$	CPU	GPU
Cantilever Beam; Edge load (0.50)	110K	26	1.37	200 s	45 s
Cantilever beam Point load (0.50)	783K	29	1.47	16 min	125 s
Knuckle (0.55)	20K	24	1.23	111 s	44 s
Bridge (0.35)	113K	32	1.45	120 s	36.2 s
Stool (0.2)	2.7M	23	1.43	84 min	14 min

the border) and distance operations are performed. One of the advantages of this method is that all of the optimization operations are performed in Cartesian grids, which makes it easier to parallelize the implementation. Method was tested using three problem cases: the

cantilever beam, MBB beam and half-wheel problem. Intel Xeon E5606 CPU and NVIDIA TESLA 2070 were used. Table 4 shows the results of speedup for MBB beam problem with different degrees of freedom. It was observed that the performance of GPU improves as the number of DoF increases.

Another work that was based on implementation of the level set method for topology optimization was conducted by Challis and Roberts [46]. However, the intermediate material densities were avoided by maintaining discrete element densities throughout the optimization process. Numerical studies for topology optimization of a base unit cube with $N \times N \times N$ dimensions (where $N = 40, 80, 120, 160$) were conducted on Intel Xeon X5620 CPUs with four possible GPU configurations: NVIDIA C2050, C2070, M2070 or S2050. It was observed that the greater the number of elements, the higher the speed up. Also, more than 90% of the computational time was spent within GPU kernels.

Ramirez-Gil et al. [15] applied TO for designing the 3D electro thermomechanical (ETM) actuators by using GPU as a co-processor. TO

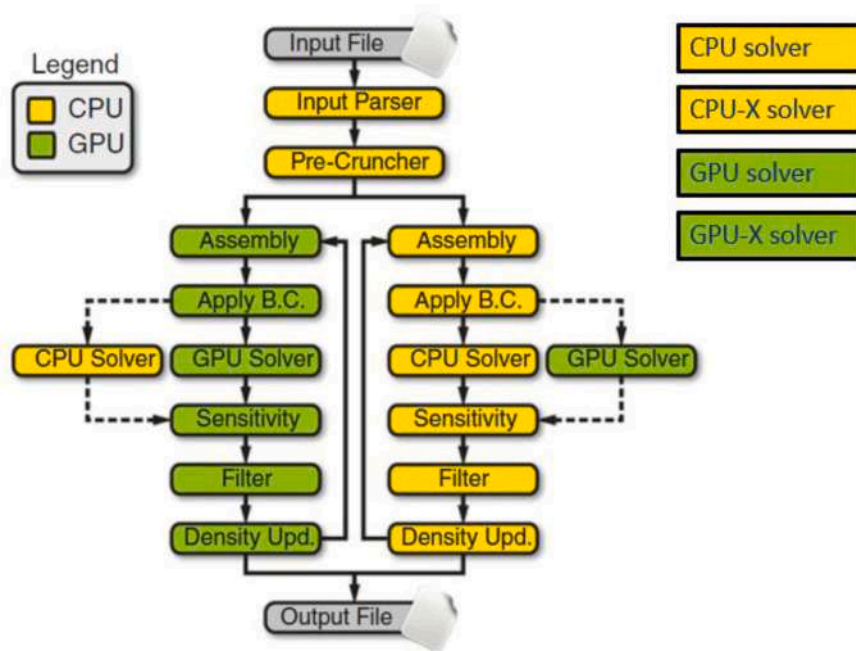


Figure 25. Schematic flowchart of the TO implementation [43]

Table 3
Runtime in seconds for three problems

Problem	CPU	CPU-X	GPU	GPU-X
Bike	62.8	274.9	267.5	56.5
MBB	192.2	809.0	792.5	186.7
MBB with hole	729.8	1,926.3	1,897.6	720.8

based on SIMP method and the algorithms were programmed in Matlab and CUDA. Figure 27 shows the TO method flowchart used in their work with code execution processors. Numerical study conducted using NVIDIA Quadro 4000 with 256 CUDA cores with CPU computations in Matlab R2013a by using Intel Xeon E5-2620. Figure 28 below illustrates

the computational time for single CPU core computation, proposed CPU-GPU implementation and speedup obtained for the latter using different mesh sizes. Proposed code shows speedup from 5x for the smaller mesh, to 35x for the larger mesh size.

Martinez-Frutos et al. [47] proposed a multi-granularity strategy for GPU computing to alleviate the problem of efficient resolution of robust topology optimization. Two levels of parallelism was adopted, data-level – to accelerate calculations required by the optimization process, and task-level – to evaluate independent simulation models. Pseudo-codes of algorithms presented with DoF-by-DoF CUDA kernel for matrix free Jacobi PCG method for FEA (dbdPCG), Element-by-element CUDA kernel for compliance and sensitivity (ebeUKU) and filtering (ebeFILTER). Two numerical studies were performed to evaluate the

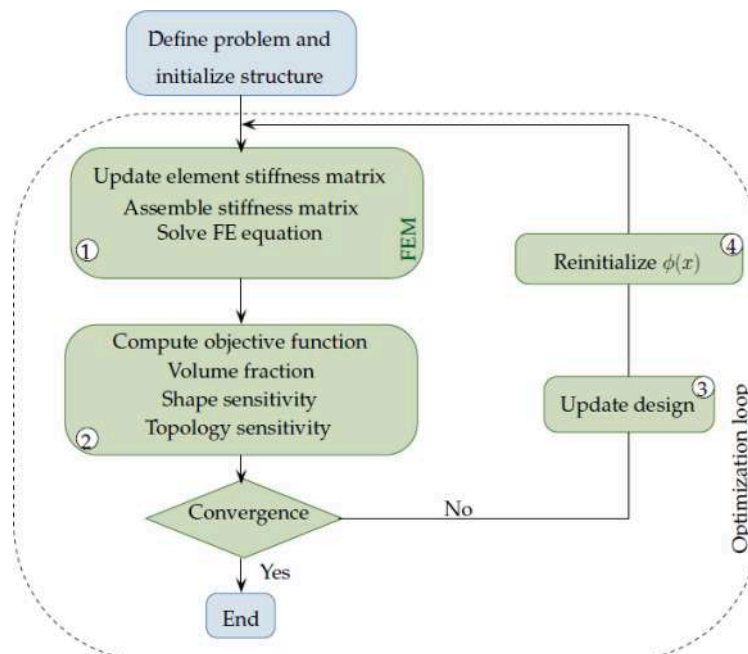
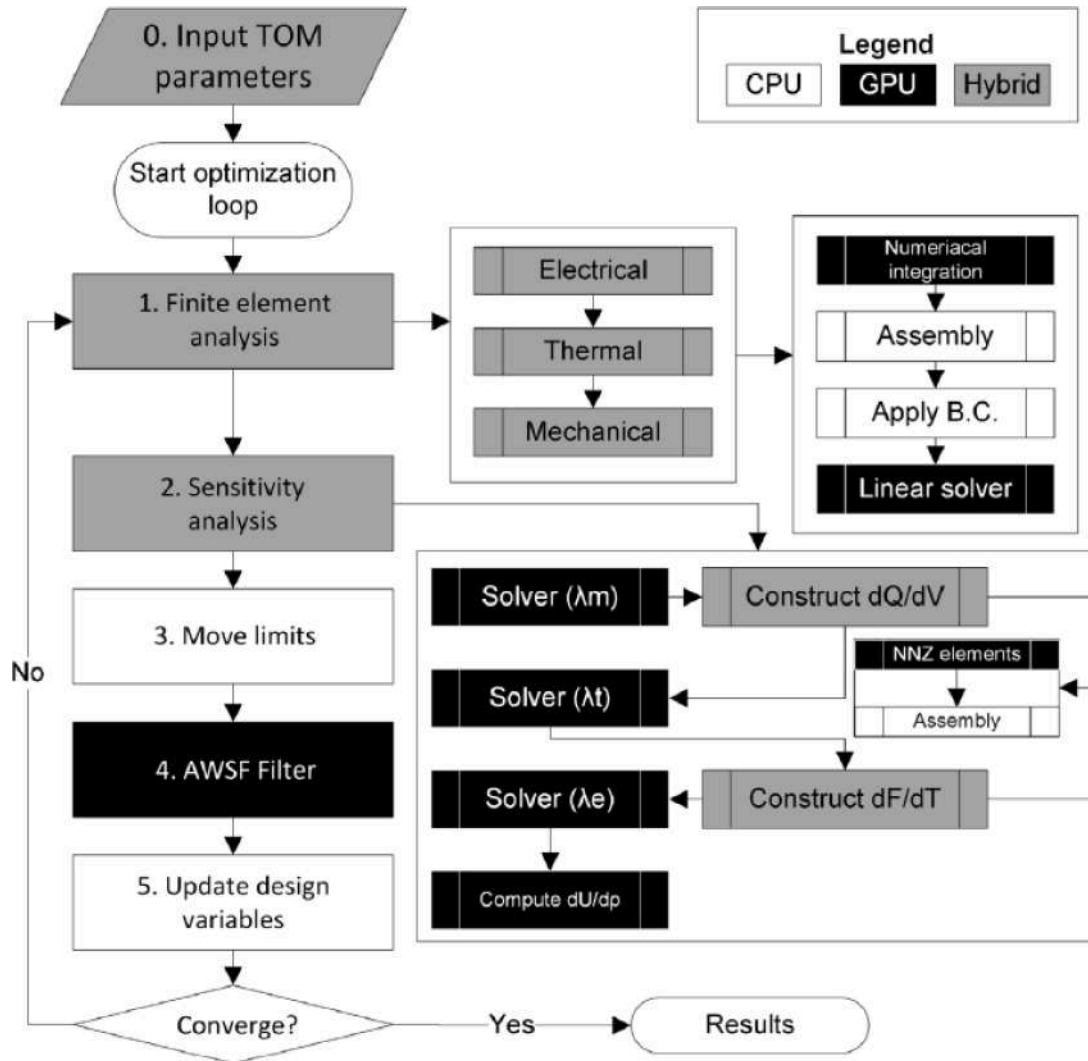


Figure 26. Flowchart of discrete implementation of TO based level set method [45]

Table 4

Speedup for one iteration of the optimization algorithm in the MBB beam problem.

n	DoF	CPU (seg)	CPU (iterations)	GPU (seg)	GPU (iterations)	Speedup $\left(\frac{CPU}{GPU}\right)$
1	28322	1.81	1472	3.42	1512	0.529
2	111938	16.07	2947	15.31	2996	1.049
3	250850	54.16	4423	49.17	4477	1.101
4	445058	132.83	5900	87.12	5963	1.525
5	694562	265.66	7366	143.112	7445	1.856
6	999362	462.18	8827	237.77	8928	1.944
9	2245538	1640.86	13244	694.33	13376	2.363
12	3989378	4160.08	17655	1574.71	17823	2.641
15	6230882	10340.1	22051	3027.59	22268	3.415
20	11072642	24608.3	29394	7444.41	29679	3.305

**Figure 27.** Flowchart of implemented TO method [15]

performance of the proposed method. First one is a robust stiffness design of a double hook structure under random load, and second is a robust heat sink design. Computer with Intel Core i7 980 and three NVIDIA GPU models: Quadro 4000 (GF100-100-KD), Tesla C2070 (GF100), Tesla K40 (GK110b) were used for the numerical study. Figure 29 illustrates the speedup results of custom-developed GPU kernels. It can be noticed how much speedup was obtained for each algorithm by using the GPU acceleration for each benchmark case.

In their other study Martinez-Fruto and Herrero-Perez [48] proposed a multi-granular GPU implementation for TO method driven by

isosurfaces. Matrix-free Jacobi PCG method was used to solve fixed grid finite element analysis (FGFEA). Figure 30 illustrates the flowchart of the optimization process and steps where GPU is used. Unlike in the previous study [47] there were presented additional Node-by-Node von Mises stress calculation (nbnStress) and volume fraction calculation (ebeVol) algorithms. Numerical studies were performed on the same workstation. Performance was evaluated using three real-life topology problems: tied-arch Bridge, MBB aircraft beam and electric mast. Figure 31 shows the speedup results for two of these cases. It can be observed that proposed GPU instance are accelerated significantly

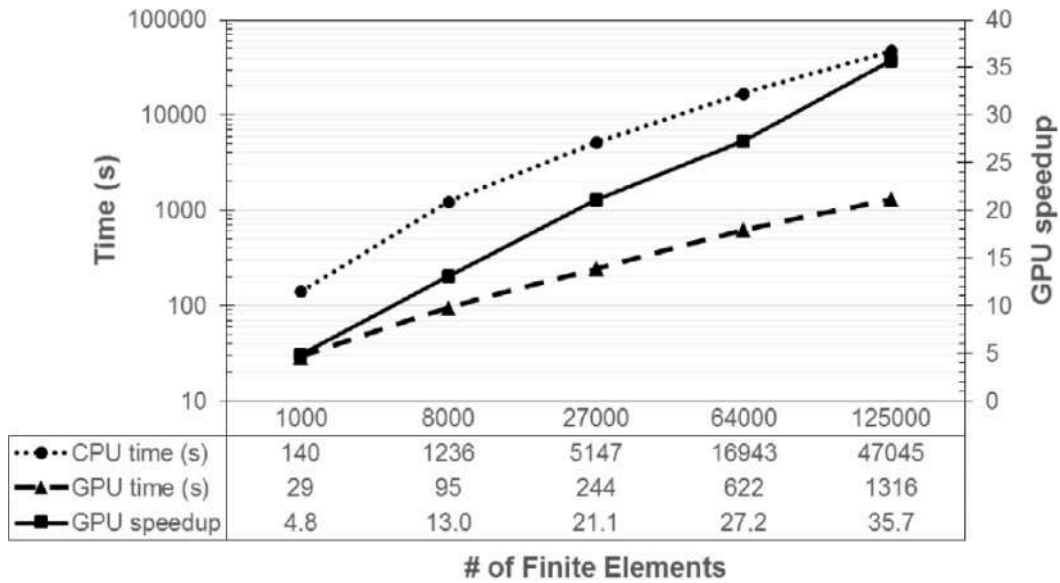


Figure 28. Computational time and GPU speedup [15]

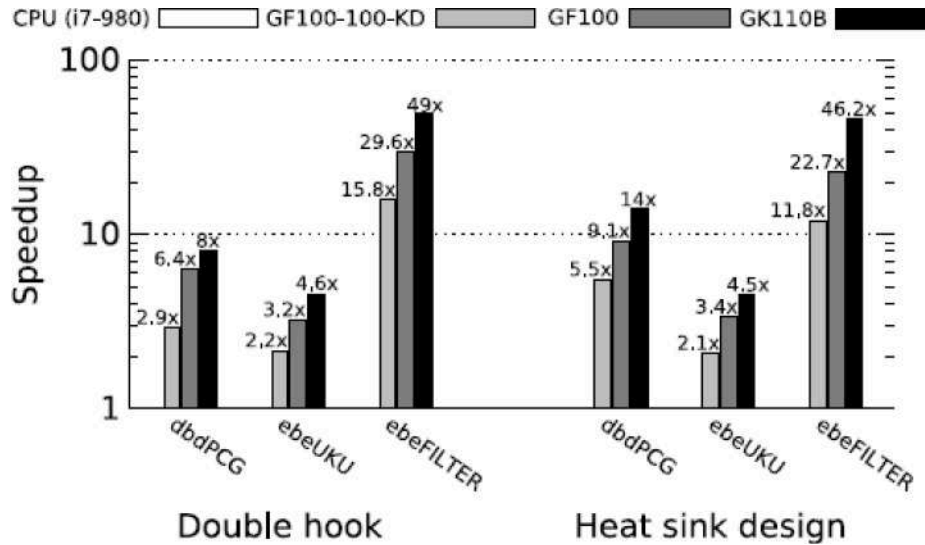


Figure 29. Speedups achieved by custom-developed CUDA kernels [47]

compared to the classical CPU implementation. Additionally, authors stated that the speedup of the GPU depends on the parallel capabilities of the graphics card.

Later same authors [49] proposed GPU implementation of the SIMP method. The flowchart of GPU instance of SIMP method is shown in Figure 32. In addition to the previous works, the algorithm of density updating strategy (ebeOC) was given in the paper.

Xia et al. [50] studied GPU parallel computing of TO, which is based on a parametrized level set method (LSM) and isogeometric analysis (IGA). The authors explained the choice of parametrized LSM for TO by indicating to its advantages in overcoming numerical shortcomings such as stability-limited computational time steps due to Courant-Friedrichs-Lewy (CFL) condition and the solution of complicated Hamilton-Jacobi partial differential equations. Furthermore, IGA based on non-uniform rational B-splines (NURBS) was considered in the TO for obtaining more accurate results compared to conventional FEM. Thus, the proposed TO method was tested for three different problems, namely cantilever beam, MBB beam, and quarter annulus shape in 2D numerical domain under various mesh sizes ranging from 32×16 up to

1024×512 . The machine was equipped with the following hardware: Intel Xeon E5-2630 v2 2.60 GHz, 16 GB RAM, and NVIDIA Quadro K5000 with 8 streaming multiprocessors and 1536 CUDA cores. The parallel computing was conducted by means of a hybrid programming model which includes MATLAB, C and CUDA. The parallelism was carried out over four stages within the algorithm: initial design domain, isometric analysis, sensitivity analysis, and update scheme. Considering the cantilever beam problem, the speedup of the CUDA was compared with CPU-based MATLAB, C and MATLAB with built in parallel processing (MP). The highest speedup of CUDA, around 700, could be archived compared to CPU-based MATLAB. The studies related to MBB beam was mostly towards evaluating the mesh size on the optimal topology, while quarter annulus results were obtained in CPU and GPU based approaches to compare their relative error values.

Gavranovic et al. [51] also used matrix-free method for solving system of linear equations with SIMP approach. Computational domain discretized with a regular structured hexahedral mesh. To improve the accuracy of the non-conformal discretization, the Dirichlet boundary conditions were imposed in a weak form using Nitsche method.

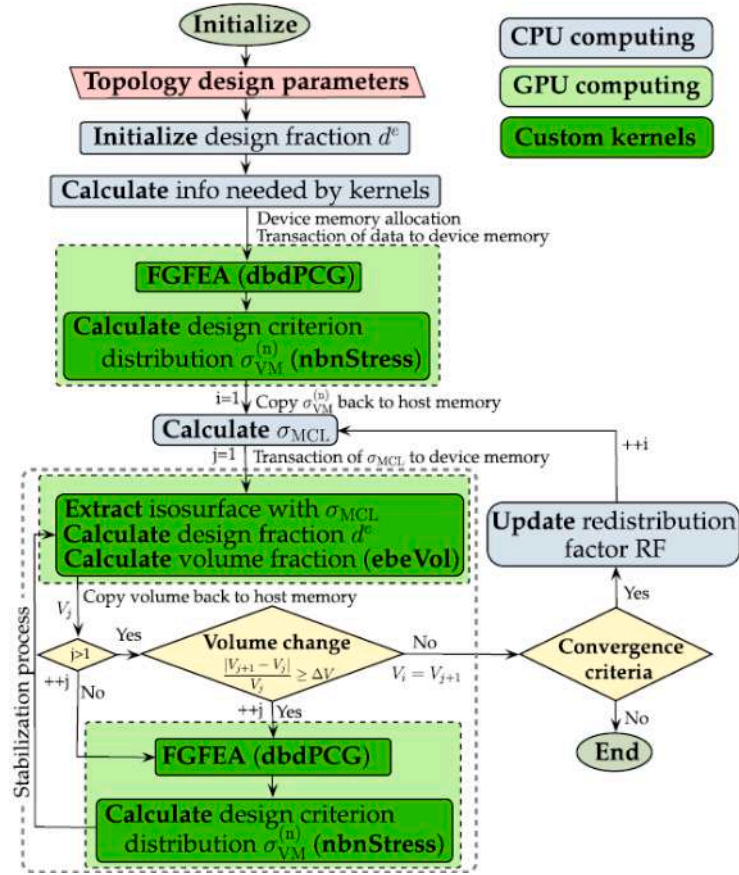


Figure 30. Flowchart of the structural optimization process [48]

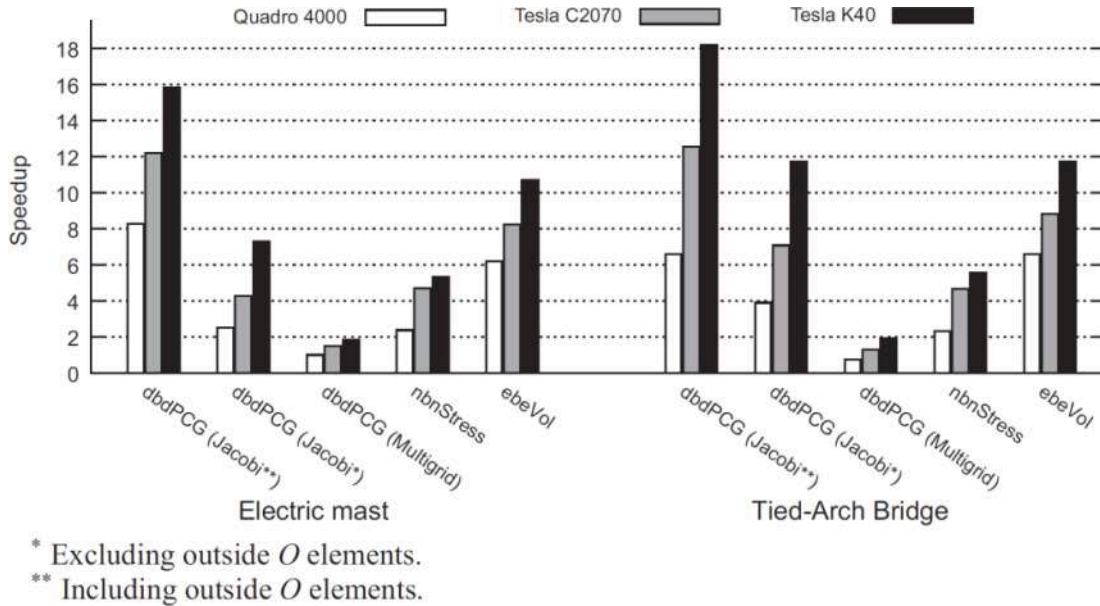


Figure 31. Speedups for benchmarks [48]

Numerical studies were performed on a machine with Intel Core I7-4710HQ with NVIDIA GeForce GTX 860M with 640 CUDA cores. 3D studies for optimization of jet bracket, two types of bridges, and cantilever beam were conducted. Simulation time was then compared with previous works, showing a good decrease in time.

In their paper Li et al. [52] proposed several accelerating strategies

for the numerical methods of topology optimization in 2D. The novelty of this research was in using so-called “two-grid strategy”. The idea was to use coarse and refined grids. Using coarse grid, finite element method was used to solve state (elastic) equation. On the other hand, applying refined grid, finite difference method was used to solve Hamilton Jacobi equation. During the process some parameters and variables were

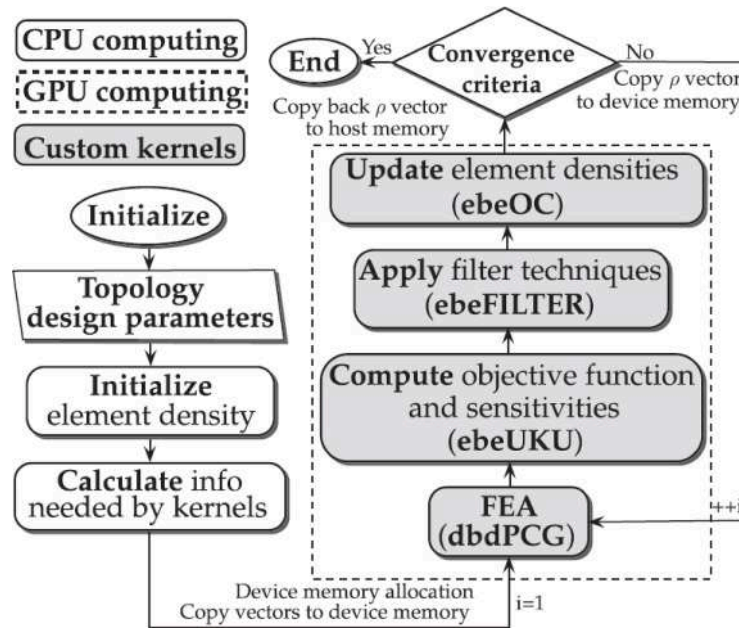


Figure 32. Flowchart of GPU instance SIMP method [49]

transferred between those two equations using the linear restricted operator. Intel I5-7200U together with NVIDIA GeForce 930MX were used on that study. 2D cantilever and MBB beams were used as a benchmark test. Figure 33 and Figure 34 represents models during different iteration for one-grid (upper row) and two-grid methods (bottom row). It was observed that proposed GPU-based two-grid schemes run up to 20 times faster without losing numerical accuracy compared with the finite element calculations.

Ramírez-Gil et al. [17] applied topology optimization to an example problem – electro-thermo-mechanical (ETM) microactuator. Since such Multiphysics problems require large computing power, the authors parallelized their sequential algorithm written in MATLAB firstly for CPU and then for GPU multicores. It was explained that voltage distribution requires electrical studies that results in Joule heating for temperature distribution evaluation, which in turn results in thermal loads that needs elastic deformation analysis. Considering that these physics problems are linear, and deformations are small, sequential solution is feasible in above-described order. The parallelization of the algorithm was conducted using Parallel Computing Toolbox (PCT) which contains of high-level functions for parallel computing using CPU and GPU. An explicit CPU parallelization was implemented for CPU multicore studies, while CUDA kernels were considered for GPU studies. Three code versions: 1) serial implementation of the CPU, 2) Multicore CPU implementation and 3) Many-core GPU implementation, were written and compared. Dell T7610 workstation and NVIDIA Quadro K5000 with

1536 CUDA cores were used for simulation. Following boundary conditions were applied to the optimization of ETM microactuator with design domain Ω : a potential difference V was applied at the anchor (electrical BC); an ambient temperature was considered at the anchors while other parts were exposed to convective heat transfer (thermal BC); and displacements in vertical and horizontal directions were limited by the anchors (mechanical BC). The purpose of the study was to maximize U_{out} in the positive horizontal direction as shown in Figure 35, a.

The highest resolution was achieved when the mesh size was 640×640 , and it was limited by the memory of the workstation. Thus, the sequential algorithm could optimize this Multiphysics problem in 116 min and 9 seconds. Figure 36 illustrates the runtime results for of three code versions. As can be noticed, it took long time to run the simulation in sequential algorithm due to the complicated nature of the problem. The best speed up was achieved with 160×160 mesh size: (i) 26.6 and 3.58 times acceleration using GPU parallel studies compared to sequential and parallel CPU studies; and (ii) 9.58 times acceleration using CPU parallel computing with respect to the serial code. The authors also provided energy efficiency results. Since such studies are out of the current review's scope, interested readers can refer to the original paper [17].

To sum up, as the TO problem size increases, methods of saving computational time become more and more important. In recent years, to achieve it different approaches have been taken, and one of them is GPU acceleration. Authors proposed different approaches and solvers

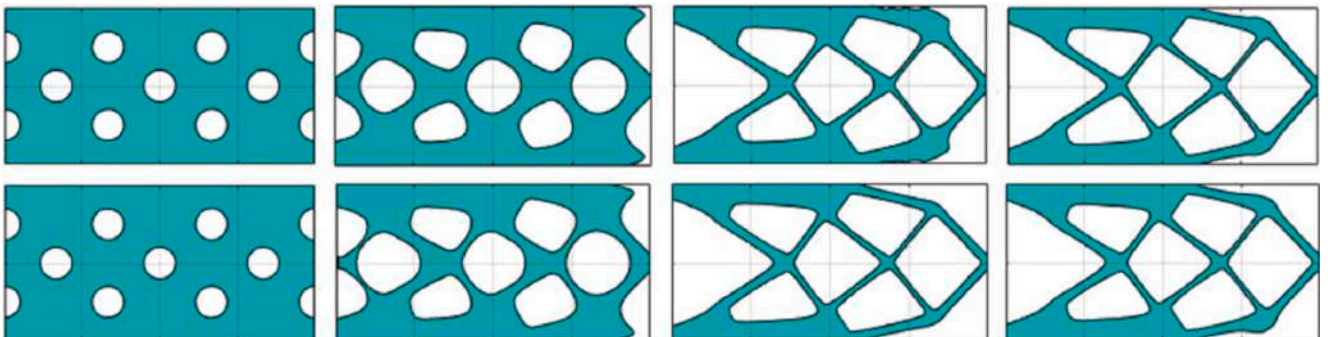


Figure 33. Cantilever model optimization with 1, 50, 150, 200 steps [52].

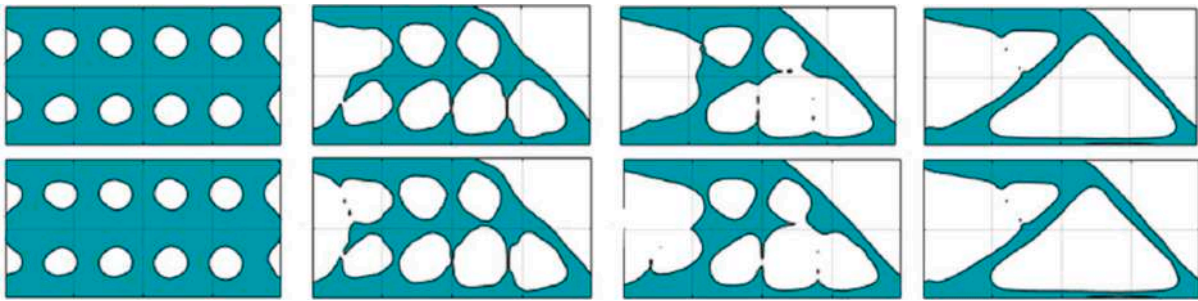


Figure 34. MBB beam model optimization with 1, 30, 80, 200 steps [52].

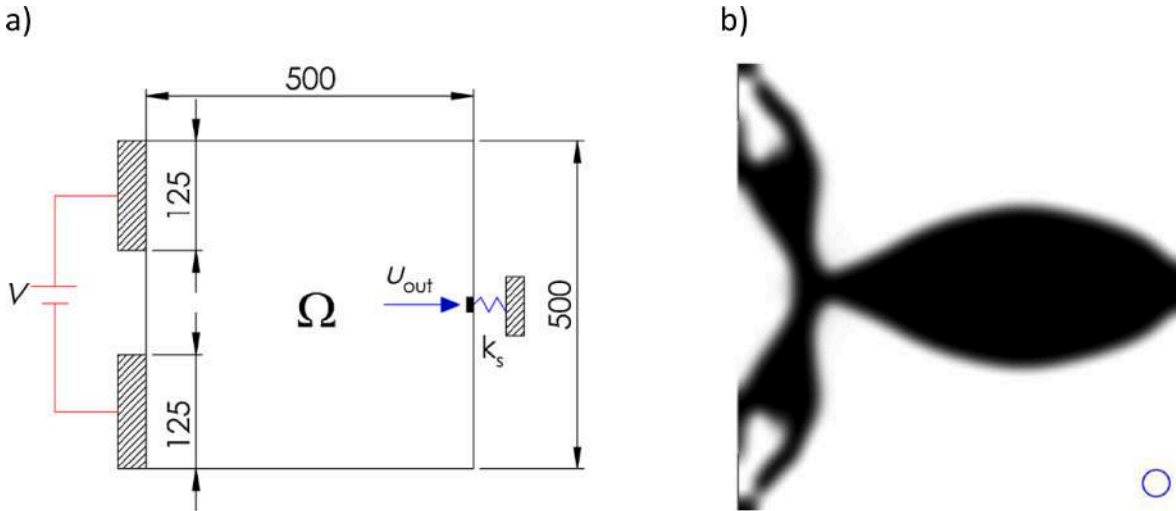


Figure 35. a) Design domain and corresponding boundary conditions; and b) Optimized actuator design using 640×640 mesh size [17]

for the implementation of TO algorithms on the GPU architectures and subsequently demonstrated the capabilities of the acceleration algorithms using different TO problems. Some authors observed that algorithm for performing the FEA is the most time consuming. Thus, some works were focused on developing an efficient solver for performing FEM analysis using GPU. Additionally, several authors proposed to use certain hybrid schemes, where computational algorithms are split between CPU and GPU. In general, it was noticed that with increasing problem size, the speedup of GPU-acceleration techniques over parallel CPU solvers increases. As can be noted from Table 5 and Figure 2, example TO problems with GPU studies included not only solid mechanics but also heat transfer and Multiphysics. Moreover, compared to CPU, GPU based TO studies are more frequent within the last decade since GPU was proven to be much more efficient for parallel computing compared to CPU.

4.3. AI-accelerated TO

With the advent of powerful computing hardware and data abundance facilitating the application of deep neural networks, researchers have been tackling various challenges in many fields, including medical diagnosis [53], robotics [54], autonomous driving [55], and opinion mining systems [56]. Deep neural networks have proven to be effective in learning highly non-linear function spaces [57, 58]. Inspired by this, in recent years scientists have started to utilize the latest data-driven deep learning algorithms to solve complex mathematical and physics problems, in which the original optimization problem is computationally expensive [59, 60]. With the need for fast and accurate structural analysis, the same methods have been applied in the field of topology optimization (TO) and have demonstrated promising results in ensuring

feasible mapping of design space with near-optimal results. Researchers tackle some of the TO problems as an image segmentation task utilizing different classes of deep neural networks. Among these models, a strong emphasis can be seen on encoder-decoder like architectures based on Convolution Neural Networks or Generative Adversarial Networks (see Table 6).

Convolution Neural Network (CNN) [61] is a type of deep neural network that has proven to be effective in various computer vision tasks. The CNN architecture consists of convolutional layers, pooling layers, fully-connected layers. CNN maps the input data to output by a series of convolutional layers, pooling layers, activation functions. Each convolutional layer performs a dot product of a convolution kernel with a corresponding input to extract the semantic feature. Moreover, an elementwise activation function is applied to the output of a convolutional layer. Pooling layers are used to control the size of the convolved layer via subsampling. Finally, a dense layer connects the final semantic information captured from the last convolutional layer to the output layer to train the network for learning the characteristic of an image for a given task. Using the ability of CNNs for capturing semantic information efficiently, encoder-decoder type convolutional networks have been developed to perform image segmentation tasks [62]. In encoder-decoder architecture, an encoder network encodes input images in a lower dimension and a decoder network learns to reconstruct images from the latent information previously encoded in the lower dimension. In autoencoder CNN architectures, this is achieved by a series of convolution and deconvolution operations to generate the target images from input images. However, very deep architectures often render the learning process particularly unfeasible as the optimization function struggles to converge. This happens due to a long chain of stacked non-linear layers in large architectures leading to

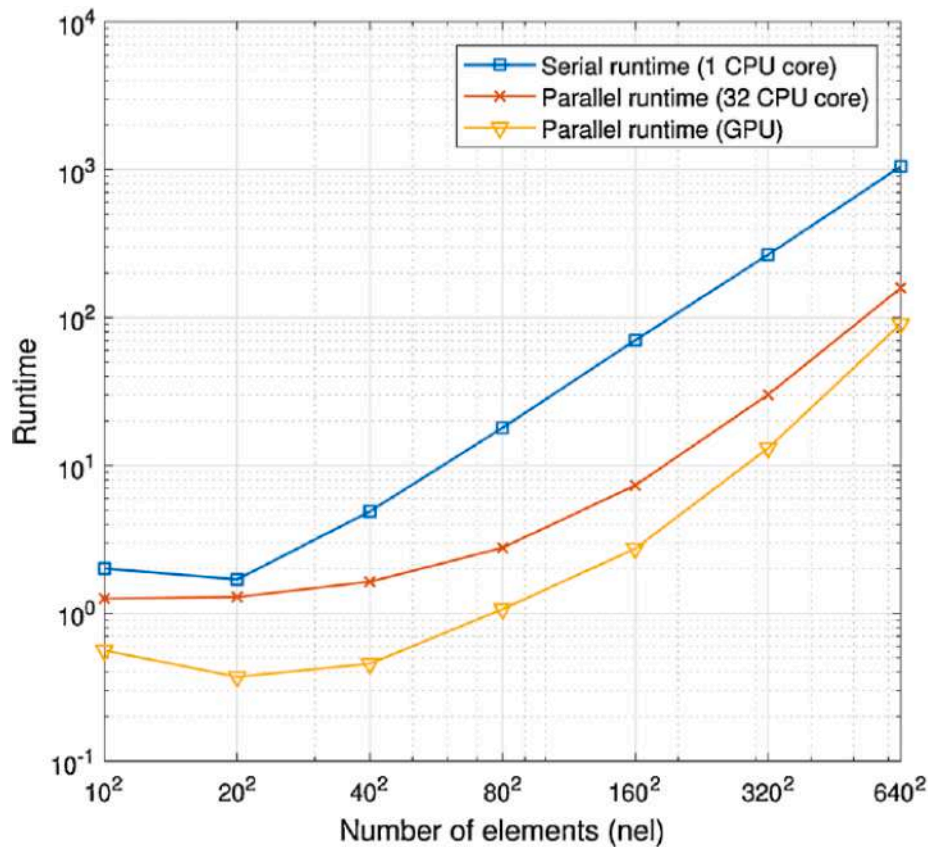


Figure 36. Computational performance result for runtime of three code versions [17]

vanishing/exploding gradients. To address this issue, He et al. [63] advised a deep residual framework to solve the gradient vanishing by utilizing skip connections in CNNs. Skips are performed by jumping over some layers in canonical order and adding the exact copy of output at some residual layer directly to the output of the next layer while skipping some layers in between. Skip connections obviate vanishing/exploding gradient problems as they enable gradients to pass through the network directly, with additional residual functions to be learned. However, to get better results in semantic segmentation with deeper architectures it's also important to capture low-level features while preserving semantic information at high levels. To achieve this, instead of using skip connections in canonical order, Ronneberger et al. [64] presented U-Net to improve the segmentation accuracy further on biomedical image segmentation. U-Net combines low-level features to corresponding high-level features by copying the former directly to the latter, allowing weight to flow through between low and high levels in a much easier way, hence achieving better results. ResUnet was introduced by Zhang [65] as a novel solution to better handle vanishing gradient and degeneration problems. The suggested framework combines both types of connections used in residual framework and U-Net. The architecture consists of residual units that are also connected through low and high levels of the network. This method eases information propagation further by the leverage of rich skip connections within the networks. Therefore, ResUnet allows us to design deeper and better networks with much fewer parameters but with superior segmentation ability.

Recent developments in the domain of generative models have led to further application of deep learning in computer vision. GAN [66] is a class of deep generative models that have recently been favoured by many researchers due to their ability to generate realistic reconstruction for the broad range of data representation and to perform relatively well on unseen data representations. GAN consists of a generator and

discriminator network that are opposed to each other during the training process. For a given dataset, the generator is trained to generate realistic data to such a degree that the discriminator is unable to discern between the real and the generated. Training continues until the discriminator is no longer able to distinguish between the real and fake data. In image segmentation, generators usually consist of CNN or convolutional encoder-decoder networks, whereas a discriminator is a conventional fully connected CNN classifier. GAN has a major drawback as it has no control over the mode of the data generator. Conditional Generative Adversarial Network (cGAN) [67] was introduced to condition the generator on additional information, thus having control over the mode of data generated. The type of the conditioning element can vary from a simple vector to a tensor.

One of such design problem domains in which the aforementioned deep learning approaches have been integrated is topology optimization. The final output of conventional topology optimization algorithms is an image representation of the optimal design structure. One of such methods is SIMP which minimizes the objective function simultaneously by changing the final material density of the design element between 0 and 1. As the resulting optimal structure is an image with values ranging between 0 and 1, it is possible to convert topology optimization from a finite element problem to a pixel-wise segmentation task. The solution is achieved by mapping initial design structures or boundary conditions to the resulting optimal structure in a supervised manner. Therefore, one has to mention that input space has to be carefully crafted and ideally has to be as much instructive as possible about the optimal target domain. Otherwise, deep learning algorithms may suffer from direct mapping when the target domain is too complex for the given input domain. Such cases happen when design space has complex displacement and external load configuration which leads to very elaborate structures that are difficult to be generalized by simple design constraints. The accuracy of the mappings may also vary based on other

Table 5

Summary of the research works on GPU accelerated TO studies

	Research work and year	Studied problem	Problem dimension	Mesh size	TO method	Hardware	Number of processors	Implementation/ solvers	Speedup
1	Wadbro and Berggren, 2009 [39]	Material distribution based on heat conduction	2D	From 56×56 to 2100×2100	Gradient-based method with a penalization technique, SIMP method	Intel Core i2 Duo, 1.86 GHz, NVidia 8800 GTX HPC: AMD Opteron 2220, 2.80 GHz	2 128 CUDA cores 2	CUDA with CUBLAS routines, preconditioned conjugate gradient solver	20 times faster than CPU 3 times faster than HPC
2	Schmidt and Shultz, 2011 [40]	Cantilever beam	3D	100×100×200	SIMP	4x AMD 6176 SE, 2.30 GHz NVidia Tesla M2050	4x 12 cores 448 CUDA cores	CUDA, matrix-free conjugate gradient solver	1.76 times faster.
3	Suresh, 2012 [41]	1) Cantilever beam 1.1) with point-load 1.2) with edge-load Knuckle Stool Table	3D	1) 32×16×16 1.1) 128×80×24 1.2) 84×28×14 20160 DOF N×N×N, N = 16-96 8×8×80	Pareto-topology optimization (PareTO) method	CPU: Intel Core i7-960 3.2 GHz RAM: 6 GB GPU: NVidia GeForce GTX 480	480 CUDA cores	Matrix free Jacobi preconditioned gradient solver	1) 6.04 sec/ 1.5 times 1.1) 125 sec/ 7.7 times 1.2) 44.8 sec/ 4.5 times Knuckle: 42sec/ 2.6 times 1.5 times for N=16, 6 times for N=96 840 sec/ 6 times
4	Zegard and Paulino, 2013 [43]	Bike frame MBB beam MBB beam with holes	2D	20378 elements 43200 elements 55200 elements	SIMP method, Optimality criteria	AMD Opteron 2216, NVidia Tesla T10	N/A	Cholesky based solver, cuBLAS and BLAS in CUDA	56.5 sec/ 4.9 times (CPU-hybrid) 186.7 sec/ 4.3 times (CPU-hybrid) 720.8 sec/ 2.7 times (CPU-hybrid)
5	Herrero et al., 2013 [45]	Cantilever Half-wheel problem MBB beam	2D	160×80 60x30 N*288 × N*48	Level set method	Intel Xeon X5620, 2.4Ghz Nvidia Tesla M2070	4 cores 488 CUDA cores	N/A	N/A
6	Challis et al., 2014 [46]	Base cell optimization	3D	N×N×N, With N=40,80, 120, 160	Level set method	Intel Xeon X5620, 2.4 GHz NVidia M2070	4 cores 448 CUDA Cores	CUDA, conjugate gradient solver	9 times for N = 40, 13 times for N = 160,
7	Ramirez et al., 2016 [15]	Electromechanical actuator	3D	Number of finite elements: 1000 – 125000	SIMP method	Intel Xeon E5-2620 2.0 GHz NVidia Quadro 4000	6 cores 256 CUDA cores	CUBLAS/CUSPARSE in CUDA, preconditioned conjugate gradient solver	from 5 times to 35 times faster
8	Martínez-Frutos and Herrero-Pérez., 2016 [47]	Double hook Heat sink	3D	96×48×120 120×60×240	SIMP method	Intel Core i7 980; Quadro 4000 Tesla C2070 Tesla K40	CPU: 6 cores	Matrix-free GPU kernels, preconditioned conjugate gradient solver	8 times faster 14 times faster
9		Cantilever beam	2D	N*32 × N*16					

(continued on next page)

Table 5 (continued)

	Research work and year	Studied problem	Problem dimension	Mesh size	TO method	Hardware	Number of processors	Implementation/ solvers	Speedup
	Xia et al., 2017 [50]				NURBS based parameterized level set method, isogeometric analysis	CPU: Intel Xeon E5-2630 v2 2.60 GHz GPU: NVIDIA Quadro K5000	CPU: built-in 6 cores GPU: 1536 CUDA cores	Compute intensive part is executed in GPU, MATLAB for matrix assembly and summation from matrices	1.44 to 691 times (CUDA vs MATLAB) 0.66 to 109.32 times (CUDA vs C) 4.11 to 145.26 times (CUDA vs MP) N/A, used to analyze influence of mesh scale on TO N/A, used for relative error estimation
		MBB beam		$N^*96 \times N^*16$					Speedup range for solving the elasticity problem for each GPU type as follows: Quadro 4000 – 2 to 8 times Tesla C2070 – 4 to 12 times Tesla K40 – 7 to 19 times
		Quarter annulus		$N^*64 \times N^*64$					
10	Martínez-Frutos and Herrero-Pérez, 2017 [48]	Electric mast	3D	$60 \times 28 \times 148$	Evolutionary TO driven by stress iso-surfaces	CPU: Intel Core i7 980 3.33 GHz GPU: Quadro 4000 Tesla C2070 Tesla K40	GPU: - 256 CUDA cores - 448 CUDA cores - 2880 CUDA cores	Preconditioned conjugate gradient using Jacobi preconditioner and geometric multigrid preconditioner	
11	Martínez-Frutos et al., 2017 [49]	Tier-arch bridge MBB beam Heat sink	3D	$184 \times 40 \times 96$ $432 \times 144 \times 144$ $128 \times 64 \times 256$	SIMP method, Optimality criteria	- CPU: Intel Core i7-5820k 3.33 GHz - GPU: Quadro 4000 - Tesla C2070 - Tesla K40	GPU: - 256 CUDA cores - 448 CUDA cores - 2880 CUDA cores	Matrix-free Preconditioned conjugate gradient using Jacobi preconditioner and geometric multigrid preconditioner	2.7 and 22.5 times
		Tied-arch bridge		$184 \times 40 \times 90$					4.1 times and 19.9 times
		Gripper		$160 \times 40 \times 80$					4.3 and 19.4 times
12	Gavranovic et al., 2019 [51]	Jet engine bracket	3D	950000 elements	SIMP method	CPU: Inlet Core i7-4710HQ • 2.5 GHz • GPU: GeForce GTX 860M	640 CUDA cores	CUDA based matrix-free multigrid method	14.4 sec
		Bridge design Another bridge design		1) 400000 elements 2) 465000 elements					1) 10.32 sec 2) 6.5 sec
		Cantilever beam		740000 elements					13.8 sec
13	Li et al., 2021 [52]	Cantilever beam	2D	1) 30×60 2) 40×80 3) 50×100 4) 100×200	Level-set method with two-grid for FEM	Inlet Core i5-7200U Nvidia GeForce 930MX	N/A	CUDA based, conjugate gradient method, Godunov's scheme	1) 0.201 / 5.5 times 2) 0.332 / 7.2 times 3) 0.545 / 8.3 times 4) 1.054 / 13.4 times 1) 0.241 / 4.9 times
		MBB beam		30×60 40×80					

(continued on next page)

Table 5 (continued)

	Research work and year	Studied problem	Problem dimension	Mesh size	TO method	Hardware	Number of processors	Implementation/ solvers	Speedup
				50×100 100×200					2) 0.354 /6.9 times 3) 0.451 / 8.8 times 4) 0.989 / 18.3 times
14	Ramírez-Gil et al., 2021 [17]	Electro-thermo-mechanical (ETM) microactuator	2D	From 10×10 to 640×640	SIMP	Intel Xeon Nvidia Quadro K5000	CPU: 16 cores GPU: 1536 CUDA cores	Parallelization of routines with GPU computing on MATLAB	25.6 times faster than serial CPU 3.58 times faster than parallel CPU

factors such as the amount of training data available and multimodality of the underlying response function. Moreover, each deep learning algorithm for any given dataset has its set of parameters that can affect the performance of the mappings.

The proposed supervised deep learning methods handle image data using convolutional layers and receive input as dense image channels (Figure 37) or discrete channels consisting of sparse matrices (Figure 38). Furthermore, although most methods rely on forming the input space mainly as image channel/matrices, some only use the scalar variables as input space (e.g., passing volume fraction as the conditional element of cGAN to generate an image from the noise). Some design parameters such as displacement vectors or stress fields can be represented in the spatial domain, whereas some scalar values such as a volume fraction do not have the characteristics of the spatial domain. However, some of these scalar variables are treated in spatial domains by creating a uniform matrix or image channel. However, in some cases, for example in encoder-decoder style networks, scalar values may also directly be passed to the latent space as extra information to support the generation (decoding).

Sosnovik and Oseledets [70] proposed an autoencoder CNN with skip connection to speed up the topology optimization process. However, instead of mapping the design constraints and conditions to the optimized structure by creating input space based on the design constraints and conditions, they proposed a rather straightforward method trying to map the intermediate density distribution to the final structure. They integrated two image channels: the first one is the intermediate design structure obtained after n th iteration of the topology optimization solver; the second represents the differences between the structure densities after the n th iteration and $(n - 1)^{\text{th}}$ iteration (Figure 39). Several models were trained on samples with various intermediate iteration numbers (n) sampled either by Poisson distribution $P(\lambda)$ or Discrete Uniform distribution $U(a, b)$, where $n \in [0, 100]$. The use of intermediate design structures here enables early stopping of the SIMP algorithm and significantly decrease total time consumption.

Lin et al. [71] adopted a similar approach to speed up topology optimization of the conductive heat transfer. Instead of using a convolutional neural network with skip connection, they utilized a type of encoder-decoder architecture similar to the U-Net. The initial iteration is conducted using the SIMP method and then its output is fed into the network to predict the high thermal conductivity. Similar to [70], topology optimization solvers is terminated at some n th intermediate iteration and its result is fed into the network to predict the high thermal conductivity. Several cases of heat conduction problem with different restricted volume fraction were experimented to analyze the robustness of the trained model Using suggested method, authors were able to significantly decrease the computation time compared to SIMP method..

Instead of using intermediate results from topology optimization methods, Zhang et al. [72] scrutinized deep CNN in an image-to-image setting to obviate extra computation by TO algorithms. For this purpose,

U-Net based encoder-decoder architecture was adopted to map informative features (i.e., image channels composed from a set of design constraints and conditions) rather than initial optimization to the target structure to accelerate topology optimization for cantilever, simply supported, and continuous beams in 2D. The proposed method was trained on image data which was extracted from design constraints, such as initial displacement field, strain field and volume fraction. Moreover, different types of residual networks were also trained on different combinations of image channels to achieve a better generalization ability. Although the final proposed method manages to provide plausible generalization ability to some extent – achieving this with 99.97% of time saved, some drawbacks such as visible structural disconnections are observed among generated samples (Figure 40). Authors suggested that this kind of problem may arise because of conventional convolution models' shortcoming on sampling from unseen observations, especially when the design space is relatively large. However, in some settings models were also observed to perform marginally better than the conventional method itself (Figure 41).

Prior to Abueidda et al. [68], the application of deep learning solutions in topology optimization was mainly confined to design areas involving linear elastic materials with little deformation. Authors conducted experiments on materials with (a) linear elasticity with small deformation (without nonlinear constraints), (b) nonlinear hyper-elasticity (neo-Hookean material) with geometric nonlinearity, and (c) linear elasticity with stress constraint, a nonlinear constraint. To achieve this, the authors employed a ResUnet architecture trained on matrix channels which are created from a set of loads, boundary conditions, and volume fractions. For each design case, different datasets were generated and separate models of the same architecture were trained. As it can be seen in Table 6, overall results suggest that predicted structures for each material case are fairly robust having very little similarity and volume fraction error.

Similarly, Kollmann et al. [73] addressed metamaterial design rather than structural systems or geometric design of materials. Authors scrutinized to reconstruct metamaterial design for three homogeneous loading objectives: (1) maximized bulk modulus, (2) maximized shear modulus, and (3) minimized the Poisson's ratio. This is done by training a ResUnet network fed on three images formed based on three optimization parameters: volume fraction, the radius of smoothing filters, and design objective. The ability of the designed method to create near-optimal structure loading objectives is strong. Yet, it has to be noted that the optimization parameters used in this work are relatively narrow and non-informative (see Table 6). Thus, it is questionable if the suggested design setting would be sufficient enough for representing real-life metamaterial designs where complex nonlinear mappings are required.

Using the advantage of generative models on unseen data, Li et al. [74] presented a non-iterative topology method based on GAN to generate near-optimal results for complex structures in high resolution.

Table 6

Summary of the research works on AI accelerated TO studies

	Research Work	Studied problem/ dimension	Mesh Size	Topology Optimization Setting	Dataset	Input Domain	Architectures tested / results		
METHODS BASED ON AUTOENCODER CNN AND U-NET									
1	Sosnovik and Oseledets, 2019 [70]	MBB-Beam in 2D	40×40	Number of nodes with fixed x and y translations: $x \sim P(\lambda=2)$, $y \sim P(\lambda=1)$ Number of loads: $P(\lambda=1)$ Value of loads: -1 Volume fraction : $N(\mu=0.5, \sigma=0.1)$	10.000 samples	Optimization result X_n obtained after the n^{th} intermediate iteration of topology optimization solver Last performed update (gradient) of the optimization results $\delta X=X_n-X_{n-1}$	SIMP with thresholding CNN $P(\lambda=5)$ CNN $P(\lambda=10)$ CNN $P(\lambda=30)$ CNN $U[1, 100]$	Binary Accuracy for 20 th iteration 97.1% 97.9% 98.4% 98.5% 98.2%	Intersection over Union (IoU) for 20 th iteration 94.3% 96% 96.09% 97% 96.4%
2	Lin <i>et al.</i> , 2018 [71]	Heat conduction problem in 2D: Volume fraction restricted to 30% Heat conduction problem in 2D: Volume fraction restricted to 40% Heat conduction problem in 2D: Volume fraction restricted to 30% and heat sink is located at the centre of the design domain	80×80	Number of heat sinks : $P(\lambda=2)$ Number of heat soruces: $P(\lambda=n/3)$, where n is the total number of grid.	10.000 samples	Optimization result obtained after the n^{th} intermediate iteration of the topology solver, where n is samled by random sampling.	Models (all design objectives trained together) CNN	Average F-2 Score (tested on a single instance) 30% restricted volume fraction case (at 30 th iteration): 90.25% 40% restricted volume fraction case (at 36 th iteration): 93.34%	30% restricted volume fraction case with heat sink in centre location (at 33 rd iteration): 91.77%
METHODS BASED ON RESIDUAL NETWORKS									
3	Zhang <i>et al.</i> , 2019[72]	Cantilever bean in 2D Simply supported beam in 2D Continuous beam in 2D	40×80	Penalty factor: 1.5 Filtering radius: 3 Design objective of beam boundary condition: one of the three given conditions Volume fraction: $U(0.2, 0.8)$ Number of forces: $U(1, 10)$ Direction of each load in x axis: X^+ , X^- Direction of each load in y axis: Y^+ , Y^-	80.000 samples	Nodal displacement on x-direction Nodal displacement on y direction Nodal strains on x-direction Nodal strains on y-direction Nodal strains on y-direction Shearing nodal stains on both x and y-direction Uniform volume fraction Shearing nodal stains on both x and y-direction Uniform volume fraction	Models (all design objectives trained together) Conventional CNN GoogLeNet U-Net U-Net with ResNet U-Net with ResNet and Dropout U-Net with ResNetU-Net with ResNet and Dropout	The pixel accuracy 81.37% 84.95% 95.47% 95.36% 95.42% 0	
4	Abueidda <i>et al.</i> , 2020 [68]	Cantilever beam in 2D: For materials with a linear elastic response under nonlinear stress Cantilever beam in 2D: For materials with linear elastic	50×50 32×32 (<i>only linear elasticity under</i>	Filtering radius: 8 cm (<i>only for materials withneo-Hookean structure</i>) Filtering radius: 12.5 cm (<i>only for linear elasticity with nonlinear stress</i>)	15.000 samples for each material case	Uniform volume fraction Nodal displacement in x-direction	Models (design objectives trained separately) ResUnet for linear elastic case under stress constraint	Dice Similarity Coefficient (DSC) 0.984	

(continued on next page)

Table 6 (continued)

Research Work	Studied problem/ dimension	Mesh Size	Topology Optimization Setting	Dataset	Input Domain	Architectures tested / results			
	response without nonlinear stress	<i>nonlinear stress</i>) Cantilever beam in 2D: For nonlinear hyperplastic response materials (neo-Hookean structures)		Filtering radius: U(3, 10) cm (only for linear elasticity without nonlinear stress)		Nodal displacement in y-direction	ResUnet for linear elastic case without stress constraint	0.958	
			Load magnitude: U(0, 15×10^4) N (only for materials with neo-Hookean structure) Load angle $\sim$ U(0, 360°) Load location: any node on the boundary of the domain		Geometric boundary conditions in x-direction Geometric boundary conditions in y-direction	ResUnet for neo-Hookean material with finite deformation		0.964	
5	Kollmann et al., 2020 [73]	Metamaterial design in 2D: maximum bulk modulus	128×128	Volume fraction: U(0.2, 0.8)	30,000 samples	Uniform volume fraction	Models (all design objectives trained together)	Mean Square Error (MSE)	DSC
		Metamaterial design in 2D: maximum shear modulus Metamaterial design in 2D: minimum Poisson's ratio		Filter radius: U(0.2, 10) μ m Design objective of metamaterial design: one of the three design cases, $id \in \{1, 2, 3\}$		Uniform filtering radius Uniform design objective: <i>id</i>	ResUnet	0.00794	0.970
METHODS BASED ON GENERATIVE ADVERSARIAL NETWORKS									
6	Li et al., 2019 [74]	Heat transfer structure in 2D	40×40 (<i>low-resolution</i>)	Number of heat skins: $P(\lambda=2)$	5,000 samples for each resolution	Heat Sink Position	Models	MSE Accuracy Score	Perceptual Hashing Accuracy
			160×160 (<i>high-resolution</i>)	Number of heat sources: $P(\lambda=n/3)$, where n is the total number of grid. Mass fraction: N ($\mu=0.3, \sigma=0.1$)		Heat Generation Position	One-phased GANnote: baseline	84.6%	80.3%
						Mass fraction	Two-Phased High-Resolution GAN	98.7%	94.8%
7	Yu et al., 2019 [69]	Cantilever beam in 2D	32×32 (<i>low-resolution</i>)	Filtering radius: 1.5	50,000 samples for each resolution	Loads in x direction	Models	MAE Accuracy Error	Mass Fraction Error
			128×128 (<i>high-resolution</i>)	Penalty factor: 3		Loads in y direction	High-Resolution GAN with	2.72%	0.51%
				Mass fraction: U(0.2, 0.8) Location of load: randomly selected any node at given the mesh Load angle: U(0, 360°)		Geometric boundary conditions in x-direction Geometric boundary conditions x-direction			
8	Rawat and M.-H. H. Shen, 2019 [77]	Cantilever beam in 2D	120×120	Volume fraction: [0.3, 0.7], increment by 0.05 Penalty factor: [2, 4], increment by 0.1	3024 samples	Volume fraction	Only the visual (qualitative) results have been discussed		
						Penalty factor			
						Filtering radius			

(continued on next page)

Table 6 (continued)

	Research Work	Studied problem/ dimension	Mesh Size	Topology Optimization Setting	Dataset	Input Domain	Architectures tested / results		
9	Rawat and M. H. H. Shen, 2019 [79]	Cantilever beams loaded in 3D	30×30×10	Filtering radius: [1.5, 3], increment by 0.1 Volume fraction: [0.3, 0.7], increment by 0.05 Penalty factor: [2, 4], increment by 0.1 Filtering radius: [1.5, 3], increment by 0.1	3024 samples	Volume fraction Penalty factor Filtering radius	Only the visual (qualitative) results have been discussed		
10	Nie et al., 2021 [80]	Cantilever beam in 2D	64×128	Penalty factor: 2 Filter Radius: 1.5 Boundary conditions: randomly selected 42 predefined scenarios Load positions: any point on the boundary of the domain Load direction: [0, π], increment by $\pi/6$ Volume fractions: [0.3, 0.5], increment by 0.02	49, 078 samples	Uniform volume fraction Nodal displacement Loads in x-direction Loads in y-direction Strain Energy Density field Von Mises Stress Field	Models note: baseline TopologyGAN with U-SE-ResNet Topology GAN with U-net TopologyGAN with SE-ResNet	MSE 0.1752 0.0599 0.0984 0.1573	MAE 0.181 0.0701 0.1014 0.1429

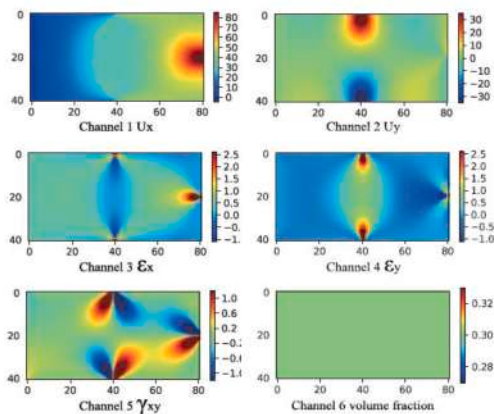


Figure 37. Input domain as dense image channels [68].

As conventional GAN models suffer from generating images at high resolution, a two-stage hierarchical refinement pipeline was proposed for predicting structures for conductive heat transfer structure. The framework was developed by coupling a GAN and a Super-Resolution Generative Adversarial Network (SRGAN) [75], where the former was used to predict low-resolution structures and the latter is trained to refine the low-resolution image by the former (Figure 42). The three boundary conditions: heat sink, heat source, and mass fraction are transformed into three matrices and fed in the deep learning model as an input. The values of the boundary conditions were sampled either from Poisson $P(\lambda)$ or Normal distributions $N(\mu, \sigma)$. For evaluating both low-resolution design generation and refinement, two datasets were created at each resolution. Initially, the first GAN is trained for generating near-optimal structures in low-resolution. The generator network of the first GAN model is a regular convolutional encoder-decoder

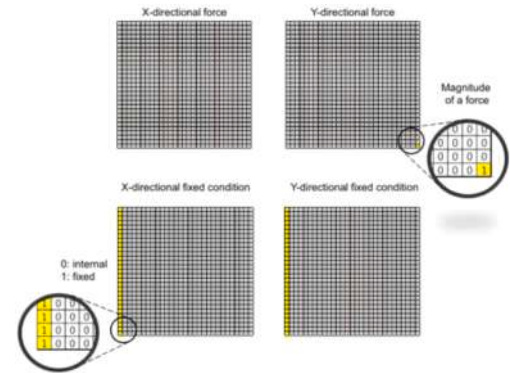


Figure 38. Input domain as sparse discretized channels [69].

network that maps the input channels to the optimized structure. The discriminator network includes seven convolution layers and two dense layers, and it discriminates the target structure at low resolution with that of generated structure at low resolution. However, low-resolution structures do not have enough spatial capacity for feasible structure design as they neglect the subtle details in the design. Thus, proposed a two-phased method where the output of the first GAN with low-resolution is integrated into the second SRGAN to refine low-resolution structures. The generator network of SRGAN is a type of residual network which uses a low-resolution image from the initial GAN to restore the most likely high-resolution image by refining it. Experiment results demonstrated that the two-phase architecture produces more detailed and robust structures compared to one-phased GAN with a direct high-resolution generation.

Yu et al. [69] devised a slightly different method to that of Li et al. [74] to replicate optimal designs for material stress at high-resolution

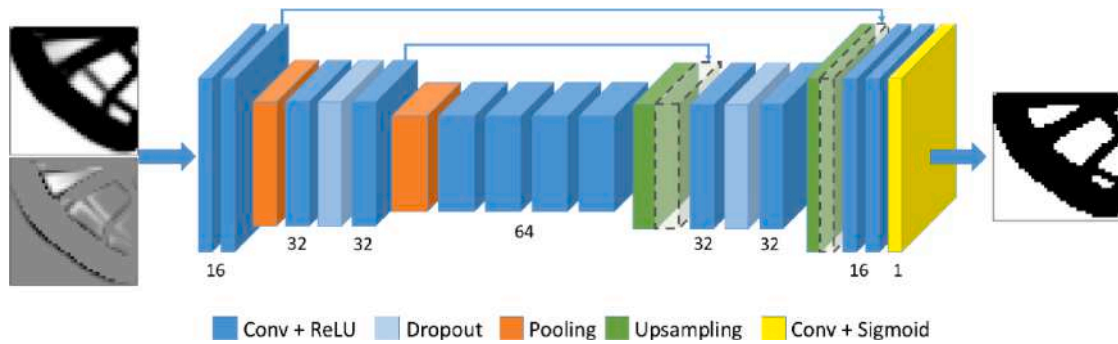


Figure 39. The architecture of the proposed neural network for topology optimization proposed by [70]. Blue arrows and opaque boxes represent the skip connections.

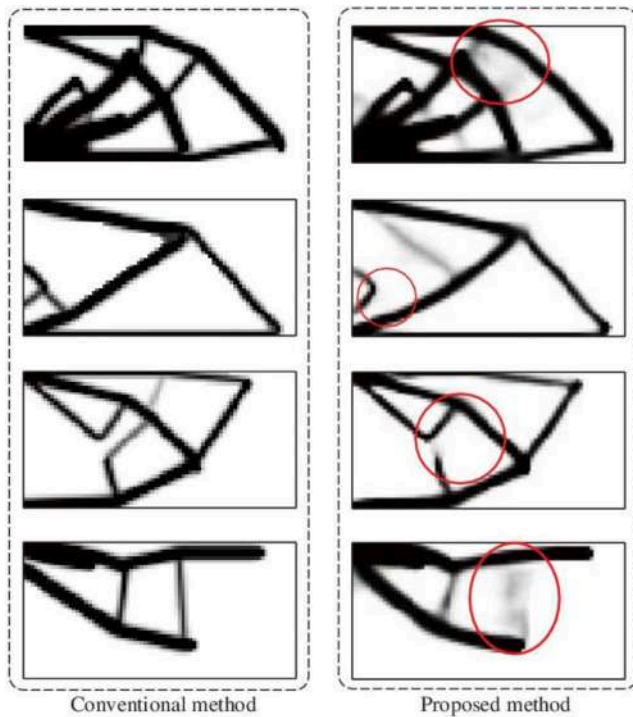


Figure 40. Examples of the results with strong structural disconnection [72]

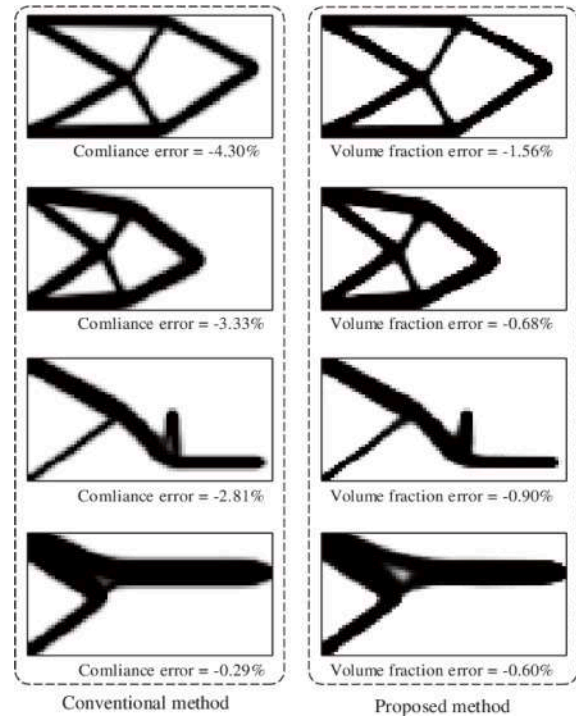


Figure 41. Example of results better than the expectation [72]

using low-resolution information. Instead of using additional GAN to integrate with SRGAN, authors directly coupled SRGAN with a convolutional encoder-decoder network that reconstructs structures at low-resolution (Figure 43). Dataset was composed of an image channel created from a set of force and geometric boundary conditions, passive zone, and mass fraction. However, as some optimization parameters are scalar (i.e., mass fractions), they are separated from the input domain and directly connected to the latent space achieved after completing encoding operation in the first network. Later, SRGAN takes the low-resolution structures generated by the convolutional encoder-decoder network and refines them to high resolution. Consequently, it tries to differentiate it from the optimized structure at high resolution.

Furthermore, the mean absolute error was added to the final loss function of cGAN. The resulting mean absolute percentage errors were 0.99% and 2.20% for train and validation sets respectively. Authors claimed that the proposed method can generate a near-optimal structure in terms of pixel values, however, some predicted structures at high resolution showed disconnections for some of the design spaces.

Although GAN has proven to be effective for generating unseen

structures, it suffers from generalizing high variability present in the data as it searches for Nash Equilibrium of the non-convex game [76]. To evade this limitation, Rawat and Shen [77] used Wasserstein Generative Adversarial Network (WGAN) [78] to predict topology optimization of quasi-optimal planar structures by conditioning it on a set of scalar design variables. Instead of choosing an image-to-image setting for image generation, authors designed their model in a rather conventional and simple manner by conditioning their model on scalar design variables and a noise channel to generate the structures. Even though being trained on a small dataset of 3024 samples, WGAN shows the ability to generate robust designs for a given single constraint of volume fraction of 0.4, while it lacks generalization quality for the other constraints.

Rawat and Shen [79] coupled the same type of conditional WGAN model employed in their previous work [77] with an additional CNN model to generate new 3D structures for cantilever beams using limited data samples. The framework was designed to generate 3D structures from the scalar design variables fed into conditional WGAN. 3D structures are more complex than 2D and hence a limited number of data samples makes training even more difficult in a 3D setting. For that, the authors used an additional CNN architecture to ensure finer mapping of the discretized 3D structures reflecting on corresponding design

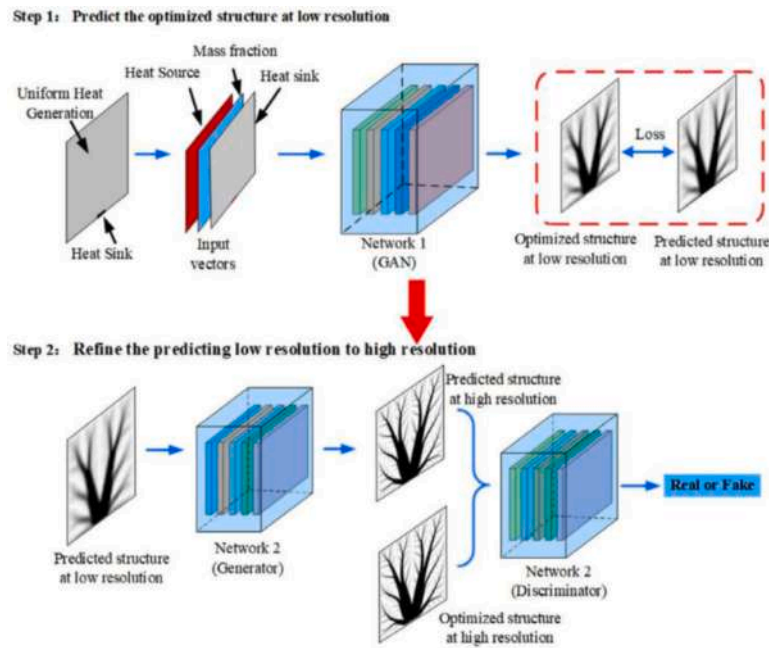


Figure 42. The illustration of the proposed two-phase method in [74]

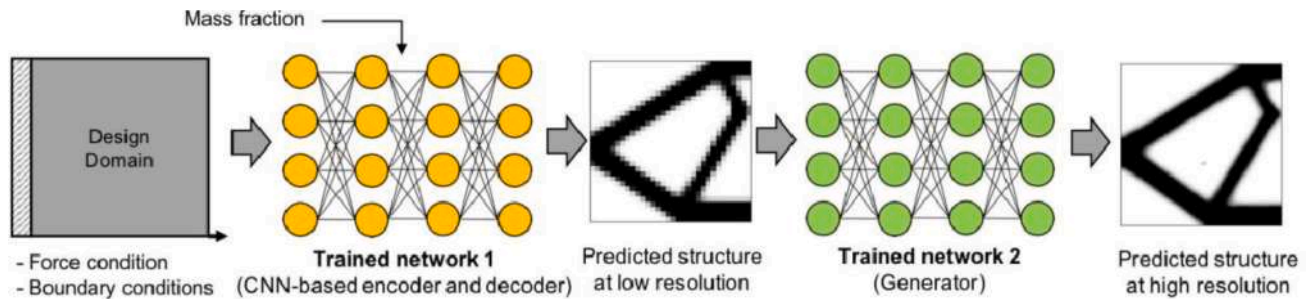


Figure 43. The conceptual flow of the proposed method in [69]

variables (i.e., volume fraction, penalty, smoothing filter). The additional CNN is incorporated to ensure that the unseen structures can be mapped to their corresponding input variables. Although cWGAN constructs reasonably optimal structures, the generated structures appear noisy because of the discretization of samples. This is addressed by

smoothing the final structure by adding a threshold filter (rounding values below 0.2 to zero) and a Gaussian filter with a kernel of 5.

Nie et al. [80] designed a strong image-to-image translation framework based on the conditional Pixel2Pixel GAN [81]. The main

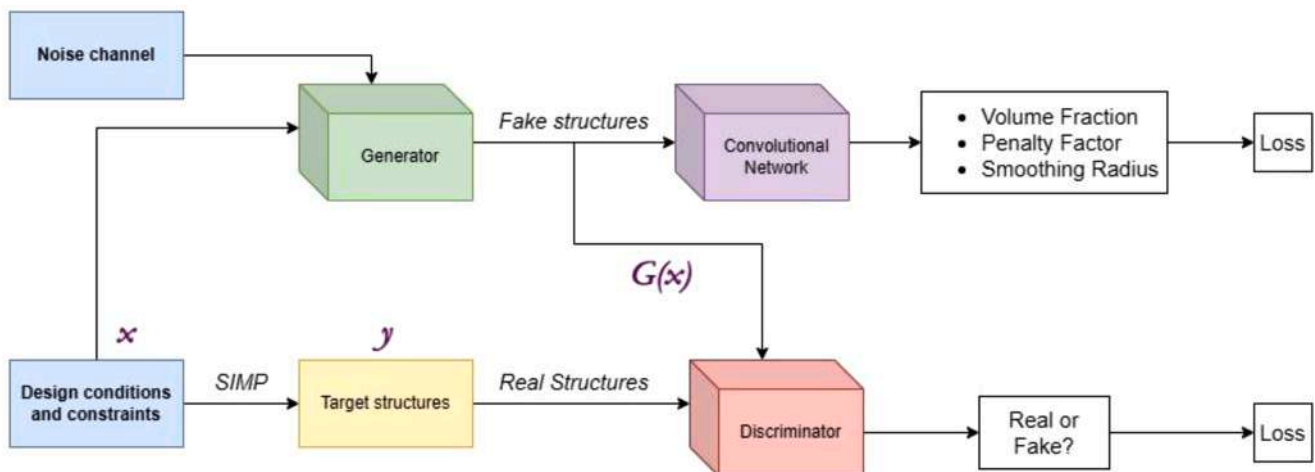


Figure 44. Semantic diagram of the proposed method by [79]

contributions made in this work were a better GAN architecture (TopologyGAN), dense input matrices involving the initial physical fields, and a generator network with a rich set of skip connections. The rich skip connection in the generator network was achieved by combining a conventional U-Net type structure with a hybrid structure formed by enhancing CNN-based ResNet [63] with SE-blocks [82]. Moreover, the author also integrated the L2 loss and the absolute error of the volume fraction to the objective loss function of the generator network to ensure that it not only maps the image to output but also evaluates mapping complying more with the design constraints. The model is conditioned on optimization variables volume fraction, different boundary conditions, load position and directions for both axes separately, Strain Energy Density field, and von Mises Stress field, giving the model very wide variability in design space. Input space formed as pixel space containing 6 image channels in total with high-density channels. As input space contains a reasonable amount of design conditions, latent space was not used in that setting. Along with different models, the author tested different combinations of image channels to find the best presentation. The final proposed method proves to be more effective than all the baseline models suggested, however, disconnections in some design structures are still overserved.

From the current trends of studies, we can observe a strong emphasis on deep learning techniques based on autoencoder CNN, U-Net, ResU-net, and GAN to accelerate TO. Although TO is becoming much more elaborate with several physics included (multiphysics) and extended to the 3D domain, studies conducted in the domain of deep learning are largely confined to studies in single physics, which is solid mechanics. Furthermore, very little attention has been given to the 3D domain (Table 6). Thus, we can conclude that the application of deep learning methods in TO is far from being mature and there are still many problems and subdomains within TO need to be tackled. Existing deep learning tools based on the aforementioned architectures have shown promising results, albeit in simple problem domains. Many of these techniques manage to construct near-optimal structures directly by mapping the initial design constraints and conditions to the target structure. Yet it has been observed that most of the suggested methods either suffer from the generalization of variability in the dataset and disconnections in the generated output, especially when the design space was highly diverse or the problem setting itself was too complex. Some of these problems that exist in the current works may perhaps be tackled by either (1) creating a better architecture or (2) utilizing a larger dataset. To create better architecture, one can seek to refine the existing deep learning topology methods or devise different methods with more promising architecture. In that direction, better encoder-decoder networks with more elaborate skip connections could be a good direction to scrutinize, as they solve gradient vanishing and degradation problems more effectively. Additionally, further contextual losses which are more related to the design problem itself can be integrated into the final objective of the deep learning models to ensure better generalization. It is also worth mentioning that, since first introduced, image-to-image translation methods have been improved significantly. Thus, employing one of the recent generic image-to-image translation models based on more elaborate GAN architecture such as StyleGAN [83], it is possible to reach better generalization ability for target structures. Furthermore, it is important to mention that instructive and extensive datasets are equally important for building powerful data-driven models. As it can be seen in Table 6, the proposed methods were trained on relatively small datasets contrary to what is required by complex data-driven models. With that in mind, large datasets can be constructed within the TO community to help to obviate the need for a good dataset for building better models.

5. Conclusion

TO area is increasing with advanced manufacturing applications. It allows to develop lightweight, durable and low cost solid structures that

can be used in single physics or multiphysics applications. Although, when it comes to the numerical implementation of TO, especially for higher design dimensions (i.e. 3D) and complicated geometries, it can be computationally expensive and time consuming. To address these problems, various numerical approaches have been used in TO studies. The current review paper deals with up-to-date research works dedicated for improving the computational efficiency of the TO. Thus, according to our studies, there are mainly three different ways to improve the computational time: CPU-based parallel computing, GPU-based parallel computing and Deep Learning. Research works related to these techniques have been reviewed and classified according to their addressed physics.

Parallel computation methods using clusters equipped with dozens of CPUs have been actively applied for TO studies starting from 2001 and this approach was the earliest among the other methods. As can be noticed from above reviewed research works in CPU-accelerated TO, most of the studies were dedicated for solid mechanics studies in 3D geometries with relatively complicated initial and boundary conditions. When domains are decomposed into subdomains for parallelization, the computational speedup versus communication between processors become challenging and various methods were considered for improvement. Moreover, several authors evaluated which parts of the algorithm need to be parallelized for faster results, while others could be skipped since the contribution of those parts such as filtering and optimization parts less than 10% of the overall computational cost. Authors considered specific structures, and tested their suggested techniques on them. Later research studies were developed towards the creation of parallel computing frameworks, which are flexible enough to study various problems such as compliance minimization (i.e. solid mechanics), and fluid flow studies. Advantages of such frameworks, users do not have to know codes and algorithms in detail yet can still carry out TO studies at an advanced level. Today, some of these frameworks are available as open source applications and software in a website named "TopOpt". Moreover, the latest works developed TO under uncertainty and stochastic approaches, which run numerical studies in CP-based parallel computing clusters.

The use of GPU-based parallel computing for TO was started to get attention from 2011 and it can be correlated with the rapid hardware and software advancement in GPU industry. Various approaches and solvers were studied for implementation in TO problems. FEM routine was noticed to be the most time consuming over the entire TO routines. Overall, it was observed that GPU-based parallel computing technique significantly reduces the simulation time. Furthermore, the speedup of GPU-acceleration over CPU-based parallel computing techniques increases as the number of elements (mesh size) in studied cases are increased. This advantage is due to the parallel processing architecture of GPU processors. Moreover, compared to CPU-based research works, the diversity of physics problems addressed in GPU studies are significant and it includes solid mechanics, heat transfer, and electromechanical studies.

In recent years, a growing interest has been observed in the application of deep learning models to TO. Many of these works approach the TO as visual imagery analysis, more specifically as an image to image translation task by utilizing the state-of-the-art deep learning models. While promising results have been achieved, proposed methods still lack the level of sophistication for being deployed to complex and elaborate industrial problems. This is largely because some suggested architectures suffer disconnections in the generated structures from having to learn the mapping of highly complex design space to the final structure with high precision. Furthermore, the current spectrum of application of deep learning methods in the field is relatively narrow as researchers have mainly focused on simple physics problems with relatively simple design space in the 2D domain. All these findings conclude that the state of studies in the field is still immature. Therefore, more in-depth research with the context of TO in mind, diverse high-quality datasets, and ever-increasing advances in deep learning may help to leverage

some of the current problems in the coming years.

From these main three areas of computational acceleration in TO it is clear that the research is overall following the trend of emerging technologies and methods. Earlier studies mainly were CPU-based acceleration, while later studies considered GPU and AI-based approaches. It is clear that CPU and GPU can further be explored in terms of applying parallel computing on new TO methods that include various mathematical formulations and computational methods, while AI-based approach is relatively new and needs more research studies in terms of choosing and developing appropriate AI and deep learning methods, their integration with TO algorithms, completeness of training data, accuracy of obtained results and other important aspects.

Declaration of Competing Interest

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

Acknowledgment

This work has been supported by the research programme of Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan (Young Researcher's Grant No. AP08856141). Moreover, the authors would like to thank Newton Funding (2015), which initiated the current international collaboration.

References

- [1] M. Attaran, The rise of 3-D printing: The advantages of additive manufacturing over traditional manufacturing, *Bus. Horiz.* 60 (5) (2017) 677–688, <https://doi.org/10.1016/j.bushor.2017.05.011>.
- [2] J.M. Flynn, A. Shokrani, S.T. Newman, V. Dhokia, Hybrid additive and subtractive machine tools - Research and industrial developments, *Int. J. Mach. Tools Manuf.* 101 (2016) 79–101, <https://doi.org/10.1016/j.ijmachtools.2015.11.007>.
- [3] C. Oztan, V. Coverstone, Utilization of additive manufacturing in hybrid rocket technology: A review, *Acta Astronaut.* 180 (January 2020) (2021) 130–140, <https://doi.org/10.1016/j.actaastro.2020.11.024>.
- [4] G. Stano, G. Percoco, Additive manufacturing aimed to soft robots fabrication: A review, *Extrem. Mech. Lett.* 42 (2021), 101079, <https://doi.org/10.1016/j.eml.2020.101079>.
- [5] M. Javaid, A. Haleem, R.P. Singh, R. Suman, Substantial capabilities of robotics in enhancing industry 4.0 implementation, *Cogn. Robot.* 1 (May) (2021) 58–75, <https://doi.org/10.1016/j.cogr.2021.06.001>.
- [6] A. Awad, S.J. Trenfield, S. Gaisford, A.W. Basit, 3D printed medicines: A new branch of digital healthcare, *Int. J. Pharm.* 548 (1) (2018) 586–596, <https://doi.org/10.1016/j.ijpharm.2018.07.024>.
- [7] J. Plocher, A. Panesar, Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures, *Mater. Des.* 183 (2019), <https://doi.org/10.1016/j.matdes.2019.108164>.
- [8] N. Morris, A. Butscher, F. Iorio, A subtractive manufacturing constraint for level set topology optimization, *Struct. Multidiscip. Optim.* 61 (4) (2020) 1573–1588, <https://doi.org/10.1007/s00158-019-02436-y>.
- [9] J. Liu, A.C. To, Topology optimization for hybrid additive-subtractive manufacturing, *Struct. Multidiscip. Optim.* 55 (4) (2017) 1281–1299, <https://doi.org/10.1007/s00158-016-1565-4>.
- [10] O. Sigmund, K. Maute, Topology optimization approaches: A comparative review, *Struct. Multidiscip. Optim.* 48 (6) (2013) 1031–1055, <https://doi.org/10.1007/s00158-013-0978-6>.
- [11] C. Lundgaard, J. Alexandersen, M. Zhou, C. Schousboe, and A. Ole, "Revisiting density-based topology optimization for fluid-structure-interaction problems," pp. 969–995, 2018.
- [12] T. Dbouk, A review about the engineering design of optimal heat transfer systems using topology optimization, *Appl. Therm. Eng.* 112 (2017) 841–854, <https://doi.org/10.1016/j.applthermaleng.2016.10.134>.
- [13] A. Neofytou, F. Yu, L. Zhang, and H. A. Kim, "Level Set Topology Optimization for Fluid-Structure Interactions," no. January, pp. 1–17, 2021, doi: 10.2514/6.2021-1686.
- [14] T. Gao, P. Xu, W. Zhang, Topology optimization of thermo-elastic structures with multiple materials under mass constraint, *Comput. Struct.* 173 (2016) 150–160, <https://doi.org/10.1016/j.compstruc.2016.06.002>.
- [15] F.J. Ramírez-Gil, E.C.N. Silva, W. Montelegrè-Rubio, Topology optimization design of 3D electrothermomechanical actuators by using GPU as a co-processor, *Comput. Methods Appl. Mech. Eng.* 302 (2016), <https://doi.org/10.1016/j.cma.2015.12.021>.
- [16] N. Aage, B.S. Lazarov, Parallel framework for topology optimization using the method of moving asymptotes, *Struct. Multidiscip. Optim.* 47 (4) (2013) 493–505, <https://doi.org/10.1007/s00158-012-0869-2>.
- [17] F.J. Ramírez-Gil, C.M. Pérez-Madrid, E.C.N. Silva, W. Montelegrè-Rubio, Parallel computing for the topology optimization method: Performance metrics and energy consumption analysis in multiphysics problems, *Sustain. Comput. Informatics Syst.* 30 (2021), <https://doi.org/10.1016/j.suscom.2020.100481>.
- [18] E.P. Paraskevoudis, K. Karayannis, P. Koumoulos, Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence, *Processes* 8 (11) (2020).
- [19] N.S. Johnson, et al., Invited review: Machine learning for materials developments in metals additive manufacturing, *Addit. Manuf.* 36 (2020), <https://doi.org/10.1016/j.addma.2020.101641>.
- [20] K. Guo, Z. Yang, C.-H. Yu, M.J. Buehler, Artificial intelligence and machine learning in design of mechanical materials, *Mater. Horizons* (2021), <https://doi.org/10.1039/d0mh01451f>.
- [21] V. Subramaniam, T. Dbouk, J.L. Harion, Topology optimization of conductive heat transfer devices: An experimental investigation, *Appl. Therm. Eng.* 131 (2018) 390–411, <https://doi.org/10.1016/j.applthermaleng.2017.12.026>.
- [22] S. Wang, "Krylov subspace methods for topology optimization on adaptive meshes," 2007.
- [23] Y. Deng, Z. Liu, P. Zhang, Y. Liu, Y. Wu, Topology optimization of unsteady incompressible Navier-Stokes flows, *J. Comput. Phys.* 230 (17) (2011) 6688–6708, <https://doi.org/10.1016/j.jcp.2011.05.004>.
- [24] O. Sigmund, A 99 line topology optimization code written in matlab, *Struct. Multidiscip. Optim.* 21 (2) (2001) 120–127, <https://doi.org/10.1007/s001580050176>.
- [25] R.E. Christiansen, O. Sigmund, Compact 200 line MATLAB code for inverse design in photonics by topology optimization: tutorial: erratum, *J. Opt. Soc. Am. B* 38 (6) (2021) 1822, <https://doi.org/10.1364/josab.427899>.
- [26] T. Borrvall, J. Petersson, Topology optimization using regularized intermediate density control, *Comput. Methods Appl. Mech. Eng.* 190 (37–38) (2001) 4911–4928, [https://doi.org/10.1016/S0045-7825\(00\)00356-X](https://doi.org/10.1016/S0045-7825(00)00356-X).
- [27] A. Evgrafov, C.J. Rupp, K. Maute, M.L. Dunn, Large-scale parallel topology optimization using a dual-primal substructuring solver, *Struct. Multidiscip. Optim.* 36 (4) (2008) 329–345, <https://doi.org/10.1007/s00158-007-0190-7>.
- [28] H. Liu, Y. Tian, H. Zong, Q. Ma, M.Y. Wang, L. Zhang, Fully parallel level set method for large-scale structural topology optimization, *Comput. Struct.* 221 (2019) 13–27, <https://doi.org/10.1016/j.compstruc.2019.05.010>.
- [29] Y. Oyanagi, "Future of supercomputing," vol. 149, pp. 147–153, 2002.
- [30] T. Borrvall, J. Petersson, Large-scale topology optimization in 3D using parallel computing, *Comput. Methods Appl. Mech. Eng.* 190 (46–47) (2001) 6201–6229, [https://doi.org/10.1016/S0045-7825\(01\)00216-X](https://doi.org/10.1016/S0045-7825(01)00216-X).
- [31] T.S. Kim, J.E. Kim, Y.Y. Kim, Parallelized structural topology optimization for eigenvalue problems, *Int. J. Solids Struct.* 41 (9–10) (2004) 2623–2641, <https://doi.org/10.1016/j.ijsolstr.2003.11.027>.
- [32] K. Vemaganti, W.E. Lawrence, Parallel methods for optimality criteria-based topology optimization, *Comput. Methods Appl. Mech. Eng.* 194 (34–35) (2005) 3637–3667, <https://doi.org/10.1016/j.cma.2004.08.008>.
- [33] A. Mahdavi, R. Balaji, M. Frecker, E.M. Mockensturm, Topology optimization of 2D continua for minimum compliance using parallel computing, *Struct. Multidiscip. Optim.* 32 (2) (2006) 121–132, <https://doi.org/10.1007/s00158-006-0006-1>.
- [34] S. Wuppapapati, A.D. Belegundu, A. Aziz, V. Agarwala, Multicriteria decision making with parallel clusters in structural topology optimization, *Adv. Eng. Softw.* 39 (5) (2008) 416–421, <https://doi.org/10.1016/j.advengsoft.2007.03.005>.
- [35] N. Aage, E. Andreassen, B.S. Lazarov, Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework, *Struct. Multidiscip. Optim.* 51 (3) (2015) 565–572, <https://doi.org/10.1007/s00158-014-1157-0>.
- [36] K. Balay S, J. Brown, K. Buschelman, V. Eijkhout, W.D. Gropp, H.D. Z, M.G. Knepley, L.C. McInnes, B.F. Smith, PETSc Users Manual - Revision 3.15, *Math. Comput. Sci. Div. Argonne Natl. Lab.* (2013).
- [37] J. Baiges, J. Martínez-Frutos, D. Herrero-Pérez, F. Otero, A. Ferrer, Large-scale stochastic topology optimization using adaptive mesh refinement and coarsening through a two-level parallelization scheme, *Comput. Methods Appl. Mech. Eng.* 343 (2019) 186–206, <https://doi.org/10.1016/j.cma.2018.08.028>.
- [38] Z.D. Zhang, et al., Topology optimization parallel-computing framework based on the inherent strain method for support structure design in laser powder-bed fusion additive manufacturing, *Int. J. Mech. Mater. Des.* 16 (4) (2020) 897–923, <https://doi.org/10.1007/s10999-020-09494-x>.
- [39] E. Wadbro, M. Berggren, Megapixel topology optimization on a graphics processing unit, *SIAM Rev.* 51 (4) (2009), <https://doi.org/10.1137/070699822>.
- [40] S. Schmidt, V. Schulz, A 2589 line topology optimization code written for the graphics card, *Comput. Vis. Sci.* 14 (6) (2011), <https://doi.org/10.1007/s00791-012-0180-1>.
- [41] K. Suresh, Efficient generation of large-scale pareto-optimal topologies, *Struct. Multidiscip. Optim.* 47 (1) (2013), <https://doi.org/10.1007/s00158-012-0807-3>.
- [42] K. Suresh, A 199-line Matlab code for Pareto-optimal tracing in topology optimization, *Struct. Multidiscip. Optim.* 42 (5) (2010), <https://doi.org/10.1007/s00158-010-0534-6>.
- [43] T. Zegard, G.H. Paulino, Toward GPU accelerated topology optimization on unstructured meshes, *Struct. Multidiscip. Optim.* 48 (3) (2013) 473–485, <https://doi.org/10.1007/s00158-013-0920-y>.
- [44] M. Planitz, "LAPACK users guide, by E. Anderson et al. Pp 235. \$19.50. 1992. ISBN 0-89871-294-7 (Society for Industrial and Applied Mathematics)," *Math. Gaz.*, vol. 79, no. 484, 1995, doi: 10.2307/3620088.

- [45] D. Herrero, J. Martínez, P. Martí, An implementation of level set based topology optimization using GPU, *Mae.Ufl.Edu* (2013).
- [46] V.J. Challis, A.P. Roberts, J.F. Grotowski, High resolution topology optimization using graphics processing units (GPUs), *Struct. Multidiscip. Optim.* 49 (2) (2014) 315–325, <https://doi.org/10.1007/s00158-013-0980-z>.
- [47] J. Martínez-Frutos, D. Herrero-Pérez, Large-scale robust topology optimization using multi-GPU systems, *Comput. Methods Appl. Mech. Eng.* 311 (2016), <https://doi.org/10.1016/j.cma.2016.08.016>.
- [48] J. Martínez-Frutos, D. Herrero-Pérez, GPU acceleration for evolutionary topology optimization of continuum structures using isosurfaces, *Comput. Struct.* 182 (2017), <https://doi.org/10.1016/j.compstruc.2016.10.018>.
- [49] J. Martínez-Frutos, P.J. Martínez-Castejón, D. Herrero-Pérez, Efficient topology optimization using GPU computing with multilevel granularity, *Adv. Eng. Softw.* 106 (2017), <https://doi.org/10.1016/j.advengsoft.2017.01.009>.
- [50] Z. Xia, Y. Wang, Q. Wang, C. Mei, GPU parallel strategy for parameterized LSM-based topology optimization using isogeometric analysis, *Struct. Multidiscip. Optim.* 56 (2) (2017), <https://doi.org/10.1007/s00158-017-1672-x>.
- [51] S. Gavranovic, D. Hartmann, U. Wever, Topology Optimization Using GPGPU, *Computational Methods in Applied Sciences* 48 (2019).
- [52] Y. Li, B. Zhou, X. Hu, A two-grid method for level-set based topology optimization with GPU-acceleration, *J. Comput. Appl. Math.* 389 (2021), <https://doi.org/10.1016/j.cam.2020.113336>.
- [53] F. Amato, A. López, E.M. Peña-Méndez, P. Vanhara, A. Hampl, J. Havel, Artificial neural networks in medical diagnosis, *Journal of Applied Biomedicine* 11 (2) (2013), <https://doi.org/10.2478/v10136-012-0031-x>.
- [54] N. Sünderhauf, et al., The limits and potentials of deep learning for robotics, *Int. J. Rob. Res.* 37 (4–5) (2018), <https://doi.org/10.1177/0278364918770733>.
- [55] S. Grigorescu, B. Trasnea, T. Cocias, G. Macesanu, A survey of deep learning techniques for autonomous driving, *J. F. Robot.* 37 (3) (2020), <https://doi.org/10.1002/rob.21918>.
- [56] S. Sun, C. Luo, J. Chen, A review of natural language processing techniques for opinion mining systems, *Inf. Fusion* 36 (2017), <https://doi.org/10.1016/j.inffus.2016.10.004>.
- [57] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” 2007, doi: 10.7551/mitpress/7503.003.0024.
- [58] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” 2013, doi: 10.1109/ICASSP.2013.6639346.
- [59] A. Bachouch, C. Huré, N. Langrené, H. Pham, Deep Neural Networks Algorithms for Stochastic Control Problems on Finite Horizon: Numerical Applications, *Methodol. Comput. Appl. Probab.* (2021), <https://doi.org/10.1007/s11009-019-09767-9>.
- [60] D. Guest, K. Cranmer, D. Whiteson, Deep learning and its application to LHC physics, *Annual Review of Nuclear and Particle Science* 68 (2018), <https://doi.org/10.1146/annurev-nucl-101917-021019>.
- [61] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.
- [62] E. Shelhamer, J. Long, T. Darrell, Fully Convolutional Networks for Semantic Segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 39 (4) (2017), <https://doi.org/10.1109/TPAMI.2016.2572683>.
- [63] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Decem*, 2016, <https://doi.org/10.1109/CVPR.2016.90>.
- [64] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9351, 2015, https://doi.org/10.1007/978-3-319-24574-4_28.
- [65] Z. Zhang, Q. Liu, Y. Wang, Road Extraction by Deep Residual U-Net, *IEEE Geosci. Remote Sens. Lett.* 15 (5) (2018), <https://doi.org/10.1109/LGRS.2018.2802944>.
- [66] I. Goodfellow, et al., Generative adversarial networks, *Commun. ACM* 63 (11) (2020), <https://doi.org/10.1145/3422622>.
- [67] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” *arXiv1411.1784v1 [cs.LG]* 6 Nov 2014 *Cond.*, 2018.
- [68] D.W. Abueidda, S. Koric, N.A. Sobh, Topology optimization of 2D structures with nonlinearities using deep learning, *Comput. Struct.* 237 (2020), <https://doi.org/10.1016/j.compstruc.2020.106283>.
- [69] Y. Yu, T. Hur, J. Jung, I.G. Jang, Deep learning for determining a near-optimal topological design without any iteration, *Struct. Multidiscip. Optim.* 59 (3) (2019), <https://doi.org/10.1007/s00158-018-2101-5>.
- [70] I. Sosnovik, I. Oseledets, Neural networks for topology optimization, *Russ. J. Numer. Anal. Math. Model.* 34 (4) (2019) 215–223, <https://doi.org/10.1515/rnam-2019-0018>.
- [71] Q. Lin, J. Hong, Z. Liu, B. Li, J. Wang, Investigation into the topology optimization for conductive heat transfer based on deep learning approach, *Int. Commun. Heat Mass Transf.* 97 (2018), <https://doi.org/10.1016/j.icheatmasstransfer.2018.07.001>.
- [72] Y. Zhang, A. Chen, B. Peng, X. Zhou, D. Wang, A deep Convolutional Neural Network for topology optimization with strong generalization ability, *ArXiv* (2019) *abs/1901.0*.
- [73] H.T. Kollmann, D.W. Abueidda, S. Koric, E. Guleryuz, N.A. Sobh, Deep learning for topology optimization of 2D metamaterials, *Mater. Des.* 196 (2020), <https://doi.org/10.1016/j.matdes.2020.109098>.
- [74] B. Li, C. Huang, X. Li, S. Zheng, J. Hong, Non-iterative structural topology optimization using deep learning, *CAD Comput. Aided Des.* 115 (2019) 172–180, <https://doi.org/10.1016/j.cad.2019.05.038>.
- [75] C. Ledig, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-Janua*, 2017, <https://doi.org/10.1109/CVPR.2017.19>.
- [76] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” 2016.
- [77] S. Rawat and M.-H. H. Shen, “A Novel Topology Optimization Approach using Conditional Deep Learning,” 2019, [Online]. Available: <http://arxiv.org/abs/1901.04859>.
- [78] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *ArXiv*, vol. *abs/1701.0*, 2017.
- [79] S. Rawat, M.H.H. Shen, Application of adversarial networks for 3d structural topology optimization, in: *SAE Technical Papers 2019-April*, 2019, <https://doi.org/10.4271/2019-01-0829> no. April.
- [80] Z. Nie, T. Lin, H. Jiang, L.B. Kara, TopologyGAN: Topology optimization using generative adversarial networks based on physical fields over the initial domain, *J. Mech. Des. Trans. ASME* 143 (3) (2021), <https://doi.org/10.1115/1.4049533>.
- [81] P. Isola, J.Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-Janua*, 2017, <https://doi.org/10.1109/CVPR.2017.632>.
- [82] J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu, Squeeze-and-Excitation Networks, *IEEE Trans. Pattern Anal. Mach. Intell.* 42 (8) (2020), <https://doi.org/10.1109/TPAMI.2019.2913372>.
- [83] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2019-June*, 2019, <https://doi.org/10.1109/CVPR.2019.00453>.