

© 2016 Danny J. Lohan

GENERATIVE DESIGN ALGORITHMS IN TOPOLOGY OPTIMIZATION
OF PASSIVE HEAT SPREADERS

BY

DANNY J. LOHAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Assistant Professor James T. Allison

ABSTRACT

This thesis explores the use of generative algorithms in engineering design. A framework for using generative algorithms in design is presented and a case study for passive heat spreaders is devised to demonstrate the execution of this framework. Topology optimization methods are now the state of the art for heat spreader design. These methods are introduced herein and are used to benchmark solutions obtained through generative design methods. The generative design methodology augments the existing topology optimization methods using evolutionary algorithms in hybrid optimization. The results presented in this thesis are the first steps in creating a rich and generalizable design optimization methodology.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to thank my adviser James T. Allison for his trust and patient support. Professor Allison has been my adviser since I entered the university my sophomore year. He has introduced me to the field of engineering design and has provided every opportunity form me to grow as a researcher. I am grateful for his guidance and look forward to continuing a fruitful collaboration.

As part of Professor Allison's research group, the Engineering System Design Laboratory (ESDL), I have had the opportunity to share time with a diverse group of becoming researchers. I would like to thank those graduate student researchers here for their continued support during my studies - Anand, Dan, Ashish, Kevin, Tinghao, Kuocheng, Jason, Lakshmi, Adam, Yong Hoon, Yashwanth, Andrew, Madhav, Satya. Working towards our research goals, we have not only shared knowledge, but warm memories of fraternity.

I am also thankful for the Industrial and Enterprise Systems Engineering department, faculty, and staff whom work together to make this degree program a entity to strive for.

Last but not least, I would like to thank the Toyota Research Institute of North America by whose generosity this research was enabled. And I particularly thank my mentor, Ercan M. Dede, whose continued support has exposed me to a breadth of engineering applications.

TABLE OF CONTENTS

| | |
|---|------|
| LIST OF TABLES | vii |
| LIST OF FIGURES | viii |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Generative Design Methodology | 1 |
| 1.2 Thesis Overview | 3 |
| CHAPTER 2 GENERATIVE ALGORITHMS | 4 |
| 2.1 Lindenmayer Systems | 6 |
| 2.2 Cellular Automaton | 8 |
| 2.3 Space Colonization Algorithm | 9 |
| 2.4 Reaction Diffusion Algorithm | 10 |
| 2.5 Generative Algorithm Categorization | 11 |
| CHAPTER 3 GENERATIVE ALGORITHM DESIGN OPTIMIZATION . . | 14 |
| 3.1 Design Parameterization | 14 |
| 3.2 Optimization Algorithms | 16 |
| CHAPTER 4 TOPOLOGY OPTIMIZATION | 22 |
| 4.1 Homogenization Method Formulation | 24 |
| 4.2 Gradient Calculation | 27 |
| 4.3 Compliance Optimization | 28 |
| 4.4 Filtering Methods | 30 |
| 4.5 Optimization Routines | 31 |
| 4.6 Homogenization Approach as a Generative Algorithm | 31 |
| CHAPTER 5 GENERATIVE ALGORITHMS FOR HEAT SPREADER DESIGN | 33 |
| 5.1 Heat Spreader Design Problem | 34 |
| 5.2 Heat Spreader Topology Optimization Review | 35 |
| 5.3 SIMP Optimization | 36 |
| 5.4 Generative Algorithm Selection | 37 |
| 5.5 Optimization Results | 40 |

| | | |
|------------|---|----|
| CHAPTER 6 | ALTERNATIVE OPTIMIZATION FORMULATIONS | 43 |
| 6.1 | Average Temperature Minimization | 44 |
| 6.2 | Max Temperature Minimization | 46 |
| 6.3 | Summary of Results | 47 |
| CHAPTER 7 | TEMPERATURE CONSTRAINTS | 51 |
| 7.1 | Temperature Constraint Variations | 52 |
| 7.2 | Numerical Case Study | 53 |
| CHAPTER 8 | CONCLUSION | 56 |
| 8.1 | Thesis Summary | 56 |
| 8.2 | Future Work | 57 |
| REFERENCES | | 59 |

LIST OF TABLES

| | | |
|-----|---|----|
| 2.1 | Categorizations of algorithms. | 12 |
| 4.1 | Penalization functions. | 26 |
| 5.1 | Generative algorithm assessment. | 38 |
| 6.1 | Computational expense of SIMP. | 48 |
| 6.2 | Summary of SIMP results. | 49 |
| 6.3 | Summary of Hybrid results. | 50 |
| 7.1 | Alternatives temperature constraint formulations. | 52 |
| 7.2 | Summary of results for different constraints | 54 |

LIST OF FIGURES

| | | |
|-----|---|----|
| 1.1 | Generative design methodology. | 2 |
| 2.1 | Generative design applications. | 4 |
| 2.2 | Types of graphs. | 5 |
| 2.3 | Generative methodology. | 5 |
| 2.4 | L-system tree example. | 6 |
| 2.5 | L-system cellular division example. | 7 |
| 2.6 | Cellular automaton truss. | 8 |
| 2.7 | Space colonization algorithm example. | 9 |
| 2.8 | Reaction diffusion example. | 10 |
| 2.9 | Algorithms by variable type. | 11 |
| 3.1 | Direct vs indirect mapping example. | 15 |
| 3.2 | Optimization algorithm comparison by class. | 16 |
| 3.3 | Unconstrained optimization example. | 17 |
| 3.4 | Patternsearch algorithm example. | 18 |
| 3.5 | Mutation operation. | 19 |
| 3.6 | Crossover operation. | 19 |
| 3.7 | Genetic algorithm example. | 20 |
| 3.8 | Sequential hybrid algorithm example. | 21 |
| 4.1 | Truss ground structure | 22 |
| 4.2 | Optimal truss designs through various optimization. | 23 |
| 4.3 | Truss with optimal topology, geometry, and size. | 24 |
| 4.4 | Homogenization method design problem. | 24 |
| 4.5 | Penalized and un-penalized topology. | 25 |
| 4.6 | Penalization functions. | 26 |
| 4.7 | Optimal topology by filtering method. | 30 |
| 4.8 | Optimal topology by solver. | 31 |
| 5.1 | Generative design methodology for heat spreader design. | 33 |
| 5.2 | Homogeneously heated design domain. | 34 |
| 5.3 | SIMP solutions for heat spreader design. | 36 |
| 5.4 | Space colonization algorithm. | 39 |
| 5.5 | Space colonization algorithm 3D example. | 39 |

| | | |
|-----|---|----|
| 5.6 | Projection of space colonization algorithm onto a regular mesh. | 40 |
| 5.7 | Compliance Optimization Results | 41 |
| 5.8 | Sample solutions obtained using hybrid approach. | 42 |
| 6.1 | Homogeneously heated design domain. | 43 |
| 6.2 | Temperature sum optimization result. | 45 |
| 6.3 | Maximum temperature optimization solutions obtained by hybrid approach. | 45 |
| 6.4 | Max temperature optimization result. | 47 |
| 6.5 | Maximum temperature minimization solutions obtained us- ing the hybrid approach. | 48 |
| 7.1 | Homogeneously heated design domain. | 51 |

CHAPTER 1

INTRODUCTION

Design for engineering is a widely researched topic today. The use of one strategy over another has large implications on the product development cycle. The products referred to here can range in complexity from consumer utensils to military aircraft. Often times engineers use intuition gained from years of experience to shorten the development cycle and make incremental progress. This is a useful strategy as product development is restricted to a finite time span. However, heuristics are developed through experience, which often requires years of practice. Automating the process of discovery for the next generation of engineering systems would save time in product development and accelerate innovation. The first steps towards this vision are presented here. In this thesis a generative design framework is proposed to challenge and discover new engineering heuristics.

1.1 Generative Design Methodology

The generative design methodology proposed has 5 distinct stages. These may be followed linearly, or may be looped for iteration. The presentation here is linear to facilitate description, Fig. 1.1.

The first stage of this methodology is to produce feasible designs that can be evaluated. This may be accomplished by obtaining existing designs from literature/practice which are proven, or automatically generating designs for future evaluation. These designs will serve as basis for learning and the development of an algorithm for targeted design optimization. The next stage of this process is evaluation. This may include physics based simulation for simple systems (e.g. convective heat transfer), or experimentation for phenomena that can not yet be simulated well (e.g. rheologically complex fluids). The evaluation is used to produce a single parameter to capture the

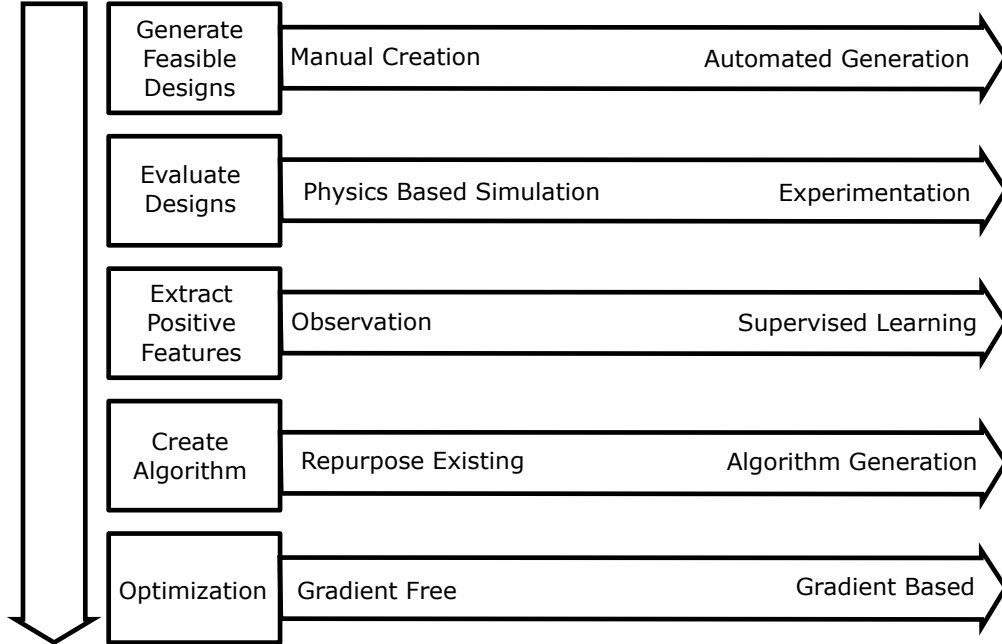


Figure 1.1: Generative design methodology.

performance of the system. This parameter is passed on to the next stage, feature extraction. In this stage, characteristics of the feasible designs are compared using the performance metric. Simple observations can be made to extract rules, such as minimizing the gap between wires in circuits to minimize inductance, or machine learning can be used to identify features which are not obvious. These features can be used in the next stage for algorithm creation. Algorithm creation does not limit design to new algorithms, but also includes existing algorithms that can be used to mimic these features, such as cellular division in truss design [1]. As an alternative, Hidden Markov Models may be used to automatically develop rules and their execution, this has been done in language processing to describe patterns that emerge in sentences [2]. The generative algorithms can then be used in optimization, where they can be tuned to target patterns that will likely increase performance. These extracted patterns may form the basis of future engineering design heuristics.

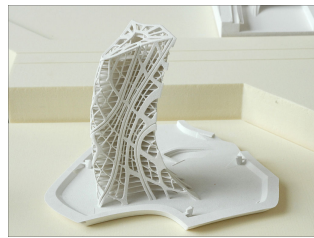
1.2 Thesis Overview

In this document, the generative design methodology is used to create a targeted design tool for the optimization of passive thermally conductive heat spreaders. First, a discussion of generative algorithms will follow with the presentation of a framework for classification. A short review of design and optimization will be presented to provide the background knowledge to understand subsequent studies. Topology optimization methods will then be presented, which are a maturing area of generative design. The next section will present a case study to benchmark the use of the generative design methodology for passive heat spreader design. An additional exploration of optimal design structure is conducted in Chapter 6, where alternative formulations for topology optimization are investigated. The final chapter will conclude this initial exploration and recommend a workflow for continuing the automation of future investigations.

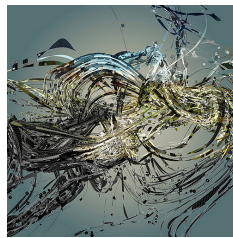
CHAPTER 2

GENERATIVE ALGORITHMS

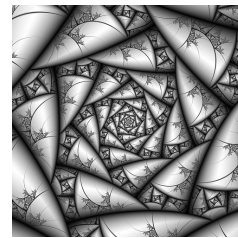
Generative algorithms are powerful tools that can be used to create or modify an object. These are used frequently in art and architecture applications to automate the development of unique structures and artforms. Consider the three examples in Fig. 2.1. These are produced by algorithms which are recursive, and hence generative in nature. In this thesis, the operations that are repeated to automate development or modification of a pattern are classified as the generative algorithm. In this framework, generative algorithms begin with a set of instructions. These are carried out for each iteration resulting in a new set of instructions.



Wikipedia commons



flickr.com/fractal_ken



flickr.com/martkknol

Figure 2.1: Generative design applications.

Visualizing a generative algorithm becomes clear when considering a generative algorithm in the context of graphs. A generative algorithm begins from an initial graph. A graph is defined by a set of nodes and edges [3]. The nodes can be homogeneous, where they are all identical, or heterogeneous where they may be different. The edges of a graph have more variability. Edges may be undirected, directed, or labeled. Several variations of these are demonstrated in Fig. 2.2.

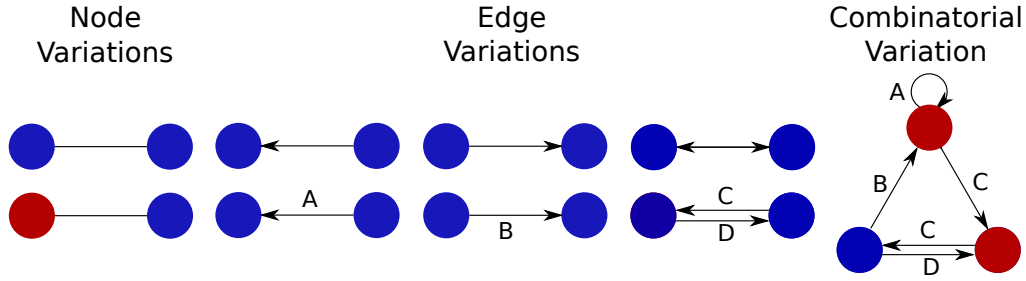


Figure 2.2: Types of graphs.

This graph is modified based on the rules of the generative algorithm to change the graph. These operations may include adding, subtracting, or modifying the parts of the graph edges, nodes, and/or labels. The resultant graph may be interpreted as the set of instructions that will produce the desired pattern. These are the most fundamental steps of the generative algorithm.

Adapting a generative algorithm to accomplish a goal, however, takes more creativity. Following the graph modification, the graph is used to build the desired pattern or structure. It is important to note that the same generative algorithm can be used to design a bridge or create abstract artwork. On the other hand, the same bridge or artwork can be obtained using different generative algorithms. Understanding this concept is important moving forward because a generative algorithm is an abstraction. It is the middle ground between a graph and final design. The interpretation of this graph abstraction may include all nodes of the graph, a particular sequence of the graph, or the final nodes of the graph. The chosen sequence of the graph is then used to produce the final pattern. Capturing this discussion in a figure, a proposed framework for interpreting generative design is presented in Fig. 2.3.

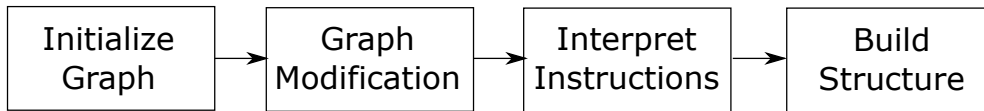


Figure 2.3: Generative methodology.

To clarify this framework, several examples of generative algorithms are presented in the following sections. These algorithms are presented with an increasing order. As a first example, Lindenmayer Systems are presented.

2.1 Lindenmayer Systems

Lindenmayer System (L-Systems) were popularized by Aristad Lindenmayer [4]. These algorithms are defined by following grammar rules recursively to create a design. These rules are usually defined a priori, but can be modified during execution. Several ways to use L-Systems will be presented to demonstrate the power of such an algorithm. Consider the following two rules for the subsequent examples,

$$[A \rightarrow B] \quad \text{and} \quad [B \rightarrow AB] \quad (2.1)$$

Venation

A simple example of an L-system is that for growing tree-like structures. Beginning with an initial node A, the system will change size as the rules are executed. The single node at the first iteration results in 5 nodes in the final iteration. To achieve a tree-like structure, consider the development of a heterogeneous non-directed graph throughout all iterations as shown in Fig. 2.4. This resembles a tree in structure directly in its' graph form. As an additional level of complexity, each node label is considered an alphabet which has an additional set of rules. These can be used to fully parameterize the final design.

$$\text{alphabet} = [\text{Length} , \text{Angle} , \text{Thickness} , \text{Label}] \quad (2.2)$$

This alphabet is attached to a node, and represents a set of instructions that describe the geometry of a line in 2 dimensions. Using the instructions at each node, a detailed branching pattern may be developed. This is demonstrated in Fig. 2.4.

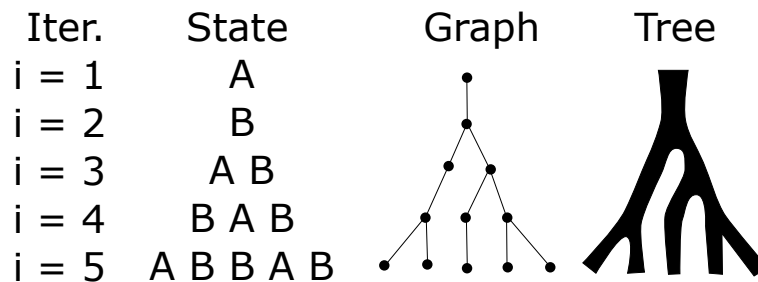


Figure 2.4: L-system tree example.

The procedure described in Fig. 2.3 was followed here. The graph was *initialized* with a single node. The two rules presented earlier were recursively followed to *modify* the graph and produce a set of A's and B's. This was then *interpreted* as a heterogeneous undirected graph. Finally, the graph was mapped to *build* a tree-like image.

Cellular Division

Earlier in this chapter it was mentioned that the same generative algorithm can be used to produce two different structures by interpreting the algorithm output differently. To demonstrate this, the same grammar rules can be used to subdivide cells. In this example the graph is no longer interpreted as the actual structure, but as a set of instructions to divide a square. Where the alphabet was previously used to parametrically define a line, now it is used to subdivide a cell in half. The graph is also now directed as to show the progression of modifications through iterations. This is performed once at every iteration for each alphabet present.

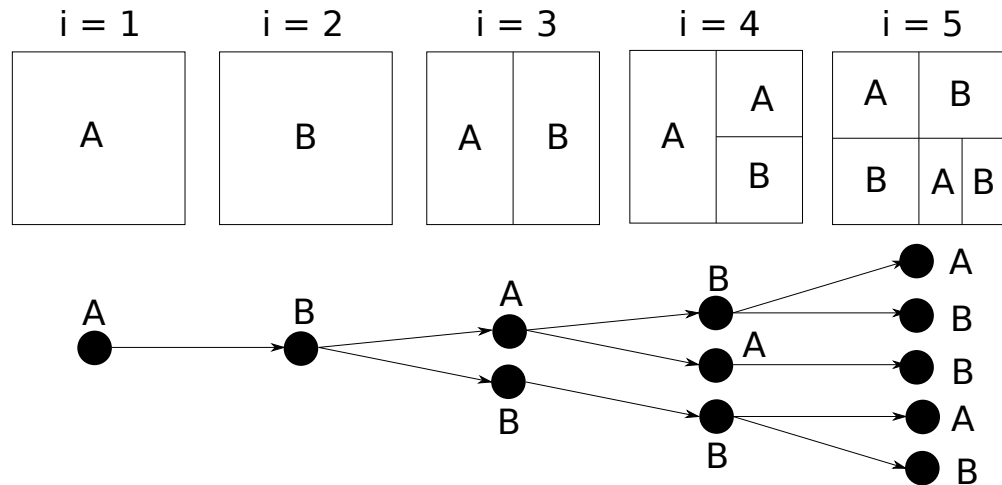


Figure 2.5: L-system cellular division example.

Once again, a graph was initialized. The same graph was obtained since the same rules were used to modify the graph. The *interpretation* of this graph has changed, where now the time history of the execution matters. These instructions were then used to *build* a subdivided square. It is important to note here that, given a set of rules, a deterministic generative algorithm will always produce the same final result. Variety in generative algorithm

output can be achieved by changing the rules, or by changing the number of algorithm iterations. For an example of the use of L-systems for the optimization of an engineering system, please refer to [1]. In this paper, L-systems were used to emulate a more complex version of cellular division to optimize a truss structure.

2.2 Cellular Automaton

The next level of complexity for generative algorithms are present in Cellular Automaton. Similar to Lindenmayer systems, cellular automaton use grammar-based rules to update a structure. This type of algorithm was studied thoroughly by Wolfram [5]. Unique to cellular automaton is the strategy of defining and applying grammar rules based on neighborhoods. These neighborhoods can be defined by adjacency, norm, or some other metric. When considering a vector where neighborhoods are defined by adjacency, the state of cell $V(j)$ may depend of the state of cells $V(j-1)$ and $V(j+1)$. For example, consider a binary vector with the following rule set:

$$[V(j) = 1 \rightarrow V(j) = 0] \quad \text{and} \quad [V(j-1) + V(j+1) \geq 1 \rightarrow V(j) = 1]. \quad (2.3)$$

Following these rules recursively reveals a symmetric pattern through time. By connecting fully all the cells with an ‘on’ or ‘1’ state, a structure made of triangles can be formed. This procedure is demonstrated in Fig. 2.6.

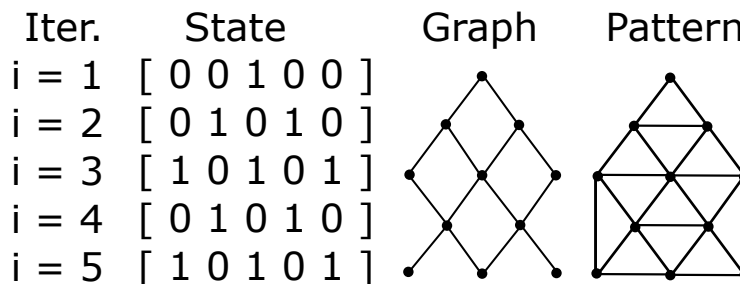


Figure 2.6: Cellular automaton truss.

The presented implementation of cellular automaton consisted of updating a one-dimensional, *initial* vector. *Modifying* this one-dimensional vector through time allows results in a final vector of the same length. *Interpreting*

these vectors in a matrix from by aggregating them reveals a geometric pattern where the 1's are nodes. Fully connecting these nodes into triangular shapes produces a stable truss structure. The same rules can be applied to spaces of higher dimension and complexity. As a famous example in two dimensions, consider “Conway’s Game of Life”, [6]. Cellular automaton was adapted for truss optimization in a paper by Khetan and Allison.

2.3 Space Colonization Algorithm

The cellular automata algorithms usually consider a discretized space that is defined prior to updating the graph. This next algorithm is similar to cellular automaton where neighborhoods are used to influence the modification of a graph, however, this is done in a continuous space. The space colonization algorithm [7] was developed to create leaf-like patterns. It does so by modeling a theory for leaf vein growth called the canalization hypothesis [8]. The canalization hypothesis suggests that leaf veins grow towards hormone centers on the leaf. This algorithm replicates this procedures by: 1) Initializing the stem, 2) Placing hormone sources on the leaf, and 3) Growing veins towards hormone sources. Though the algorithm was developed to model leaf veins, it can be adapted for other purposes. Consider the example in Fig. 5.4.

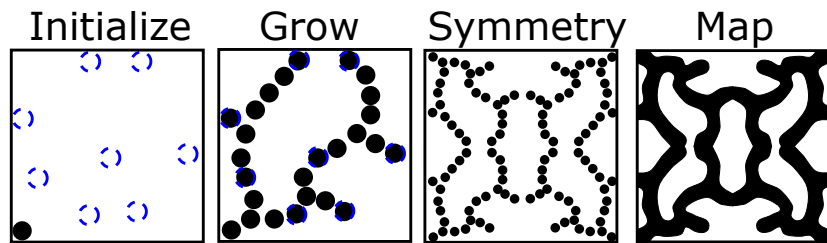


Figure 2.7: Space colonization algorithm example.

Analyzing this from a the definition of a generative algorithm presented in Fig. 2.3, the similarities becomes clear. The graph was *initialized* with a stem node and auxin locations. The graph was *modified* based on rules defined by neighborhoods. The resultant graph is *interpreted* directly as its structure represents leaf veins. The nodes of this output can be used to trace

and *build* the final structure. The algorithm produces a dendritic structure, which after symmetric reflections, changes into a new type of structure. This type of structure resembles solutions from micro-structure design in solid mechanics. The space colonization will be investigated further in subsequent chapters as a tool for designing heat spreaders.

2.4 Reaction Diffusion Algorithm

At a glance, reaction diffusion models are another example of continuous hybrid cellular automata. Reaction diffusion equations were developed to model chemical processes. They are interesting as they combine two different types of algorithms together. The first algorithm is based on continuous equations for diffusion. The second algorithm involves discrete local updates, or reactions. Consider a space filled with red (R) and black (B) particles, as shown in Fig. 2.8. The particles will move (diffuse) based on some physics-based criteria. If a certain combination of particles come within a distance of each other, then a reaction takes place. For this example consider the following grammar based reaction:



When two red particles come in contact with a blue particle, the red particles are changed to blue. The execution of this procedure is illustrated in

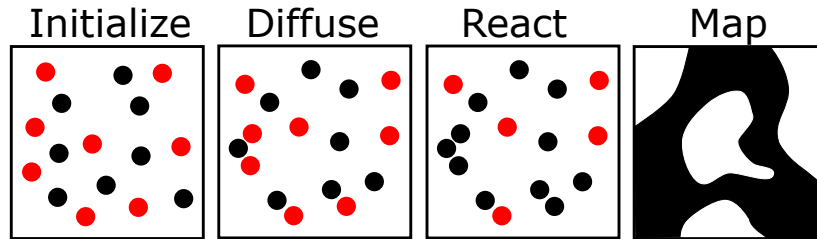


Figure 2.8: Reaction diffusion example.

Fig. 2.8, where a set of particles is initialized and allowed to diffuse for a prescribed time. A reaction may take place if the conditions are right, and this changes the composition of particles on the domain. If this set of particles is interpreted as a Voronoi type diagram, the local regions around a particle

are assigned to it, resulting the mapping shown in the figure. This can be used to generate a wide variety of interesting patterns. In practical application, the chemical domain may be discretized into finite elements and the particular amounts in each cell may be controlled. This presents a discrete domain alternative to the continuous representation.

2.5 Generative Algorithm Categorization

Generative algorithms such as the ones presented above are used in various domain-specific applications. An element that is important to generalizing the use of these algorithms is understanding the similarities and differences between the natures of the algorithms. For example, when observing the discrete or continuous properties of the algorithms, they may be interpreted as shown in the following figure:

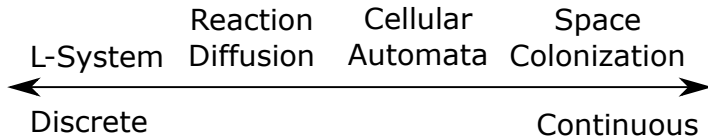


Figure 2.9: Algorithms by variable type.

The L-system algorithm is the most discrete algorithm based on its grammar rules. The reaction diffusion model may be the most continuous of the algorithms, though also has some discrete elements. The remaining two algorithms fall between these two based on this criteria. This simple example for categorization is one of many that can be used to classify generative algorithms. As step towards a deeper understanding of algorithm relationships, Table 2.1 has been compiled.

It is important to note that these algorithms can be modified to obtain characteristics that may be necessary for a specific application. The presented categorization was made considering basic forms of each algorithm for the sake of argument and comparison. The following paragraphs will describe each categorization and the placement of each respective algorithm.

Physics Update: This criteria describes algorithms that are updated based on first principles or fundamental laws of physics. A clear example of this is

Table 2.1: Categorizations of algorithms.

| Categorizations | Algorithm | | | |
|----------------------|-----------|----|----|----|
| | RD | CA | SC | LS |
| Physics Update | ○ | | ○ | |
| Neighborhood | ○ | ○ | ○ | |
| Grammar | | ○ | | ○ |
| Discrete Event | ○ | ○ | | ○ |
| Parameter Controlled | ○ | | ○ | |
| Variable Sized | | | | ○ |

| KEY | |
|-------------------------|-------------------------|
| RD = Reaction Diffusion | CA = Cellular Automaton |
| LS = L-System | SC = Space Colonization |

a reaction diffusion algorithm. These are typically used to model chemical reactions. This characteristic in an algorithm is beneficial for efficiently producing engineering designs.

Neighborhood: These algorithms are updated based on the status of local parts of the system or spatial relationships. When using cellular automata, the status of a cell is updated based on the status of its neighboring cells. The space colonization algorithm also uses local information about all auxin to define growth direction. The reaction diffusion equations update based on the local information as well.

Grammar: Grammar-based algorithms are updated using on a set of defined rules. For example the rules that define neighborhood updates when using cellular automata. L-systems also use these exclusively when the growth of the structure is chosen apriori. For this reason, L-system are not classified as neighborhood update algorithms.

Discrete Event: These algorithms will perform changes based on the occurrence of a discrete event. The L-system, cellular automaton, and reaction diffusion algorithms all have this characteristic. Examples of discrete events include the switch of a state or passing of a threshold.

Parameter Controlled: These algorithms are not defined by designed rules, but parameters that control a relationship. Clear examples of this

are the reaction diffusion and space colonization algorithms that use continuous equations to update the graph.

Variable Sized: This describes algorithms that grow a design in size during operation. The L-system algorithm is only algorithm that is not bounded by growth. Output graphs may increase in size indefinitely.

CHAPTER 3

GENERATIVE ALGORITHM DESIGN OPTIMIZATION

To understand how generative algorithms can be used in optimization, a short review of optimization is provided here. For an in-depth presentation of design optimization, please refer to Refs. [9, 10]. Consider a general optimization problem as shown in Eq. (3.1).

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && \Theta(\mathbf{x}) \\ & \text{s.t.} && \mathbf{h}(\mathbf{x}) = 0 \\ & && \mathbf{g}(\mathbf{x}) \leq 0 \end{aligned} \tag{3.1}$$

There is some objective function, $\Theta(\cdot)$, that depends on a vector of design variables \mathbf{x} . This objective may be subject to a set of equality constraints, $\mathbf{h}(\mathbf{x})$, and/or inequality constraints, $\mathbf{g}(\mathbf{x})$. How this optimization problem is formulated greatly influences whether the problem can be solved and what type of solution will be obtained. In this chapter the use of generative algorithms in optimization formulation will be clearly outlined. In addition, a high-level review of optimization algorithms used in subsequent studies will be presented.

3.1 Design Parameterization

As discussed in the previous chapter, generative algorithms are defined by the creation of a set of instructions, and the execution of these instructions to create a pattern. To use generative algorithms in design optimization, the parameters that guide the growth of the generative algorithm are used as design variables. Optimizing parameters that guide the growth of the algorithm

results in an indirect mapping for optimization, or a design representation abstraction. To understand better this mapping, consider the specification of a shaft, shown in Fig. 3.1. To define directly the shaft design representation, independent geometric dimensions may be selected. An alternative abstract design representation could involve adding one rule that increases or decreases the diameter of the shaft at each stage, and one that modifies the radius. This would form a generative algorithm that modifies design properties (as opposed to adding or subtracting elements). This design representation, however, reduces the set of achievable shafts, but provides an efficient shaft design representation in terms of optimization variable dimension.

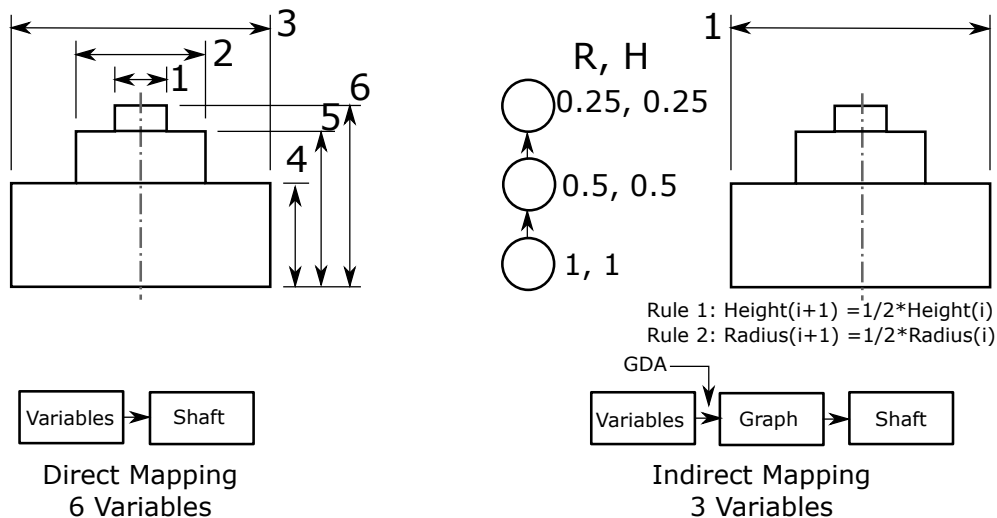


Figure 3.1: Direct vs indirect mapping example.

Direct mappings typically result in larger-dimension design problems; design decisions must be made for every detailed design element. This representation allows for maximal design flexibility, however, the design problem may become difficult to optimize with the large number of design variables. Indirect mappings have a reduced number of design variables. This often reduces design problem solution difficulty, but eliminates access to at least some designs due to dimension reduction (i.e., design space coverage is reduced). If a generative algorithm intelligently provides coverage of important regions of the design space, it can help focus design search efforts and increase the likelihood of finding better designs. An interesting example of the success of an indirect mapping for soft robotics is present in the work of Cheney et

al. [11], where indirect mappings supported the identification of meaningful designs, whereas direct mappings could not. This effect is generalizable to other systems. This will become apparent for the heat transfer application presented in this thesis.

3.2 Optimization Algorithms

The algorithms presented here can be separated into three categories: 1) Gradient-based, 2) Gradient-free, and 3) Memetic. Gradient based optimization strategies are used to locate a point where the gradient of the objective function is zero (or satisfy similar requirements for constrained problems). This point may be a maximum, minimum, or a saddle point. These algorithms move towards this stationary point using objective rate of change information. This can be calculated either analytically (exact), through adjoint techniques, finite difference methods, or other sensitivity approximation methods. Gradient-free algorithms do not use derivative information as a basis for optimization. These algorithms use different types of rules to search the design space for an optimum, usually requiring a large number of objective evaluations to converge. Memetic algorithms combine characteristics of both algorithms to speed up convergence. These will be discussed in more detail later. As a point of comparison, these classes of algorithm are compared in Fig. 3.2.

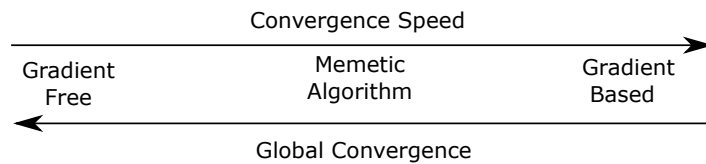


Figure 3.2: Optimization algorithm comparison by class.

As mentioned in the previous paragraph, gradient-based algorithms are typically much better at converging quickly to a local optimum than gradient free methods. However, this does not guarantee convergence to a global optimum. Though guarantees cannot be made for any algorithm to converge to a global optimum unless certain restrictive conditions are satisfied, the likelihood of finding a better optimum increases when using gradient free methods.

Gradient-Based Optimization

Consider unconstrained optimization, which refers to a problem where the objective function is in sole consideration. A fabricated test function has been developed and is presented in Fig. 3.3. The function is well defined and continuous along the solid blue line. At each iteration the gradient is calculated and used to direct the search towards the optimal solution. In this particular example, each step is damped to improve convergence. This strategy is known as a damped gradient descent [12]. The likely progression for this algorithm is presented in Fig. 3.3.

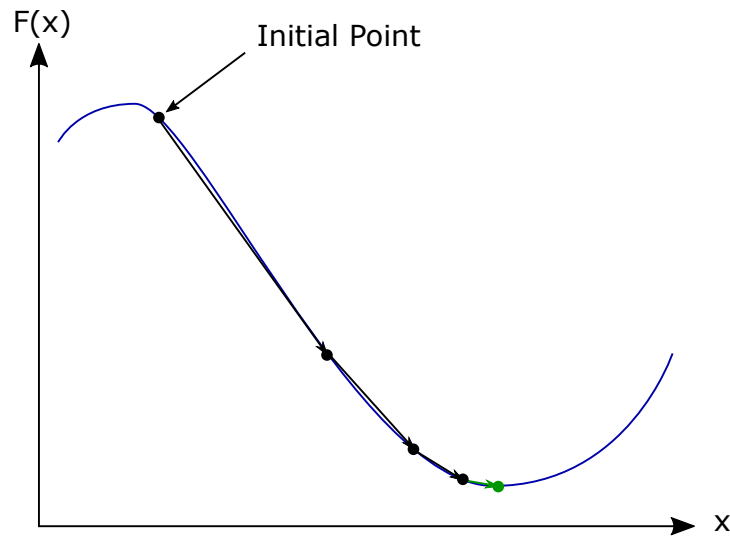


Figure 3.3: Unconstrained optimization example.

Since this class of optimization algorithm uses gradient information to move towards a local minimum, these methods typically converge in fewer steps than gradient-free methods. In this thesis, several gradient-based algorithms are investigated for design optimization. These will be described prior to their use.

Pattern Search

The pattern search algorithm is a gradient free method that falls under a class of algorithms called “direct search” algorithms. Algorithms of this type search the design space iteratively by taking the best possible step at each

iteration from a pre-defined template [13]. Like the gradient-based methods, the pattern search algorithm requires an initial starting point. From this starting point the algorithm will *poll* the design space by evaluating designs in all predefined search directions (in the example strategy presented here, two per design variable). Of the candidate steps, the pattern search algorithm will take the step that results in a design with the best objective function value¹. If none of the step directions decrease the objective function value, the step size is reduced and a new poll is taken.

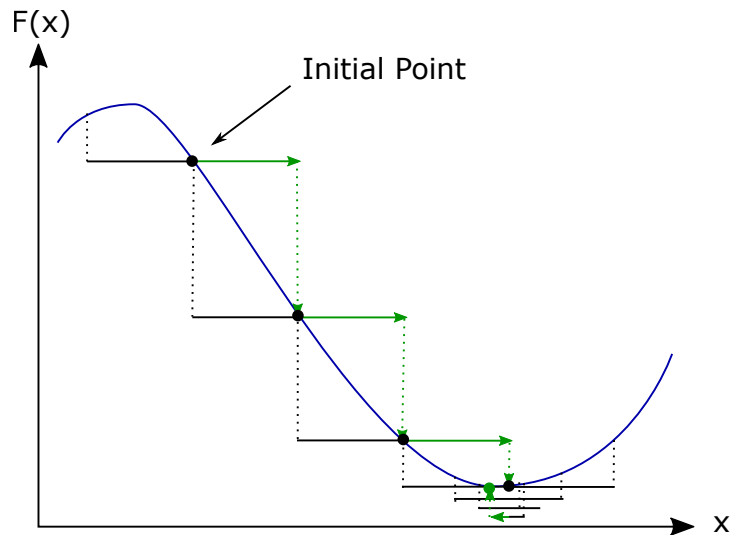


Figure 3.4: Patternsearch algorithm example.

This procedure is demonstrated in the one-dimensional example in Fig. 3.4. The algorithm converges when the poll size falls within a defined tolerance. The pattern search algorithm is less sensitive to disturbances when compared to the gradient-based methods, since the poll size may increase or decrease at a given position. It can be used to optimize non-smooth functions, but it is not a global optimization algorithm. Pattern search is a local search algorithm, and may pass over the global optimum if the poll size is too large, or if started at a point that leads to a local optimum instead of the global.

¹An alternate strategy is to select the first step that provides descent.

Genetic Algorithm

A genetic algorithm is another type of gradient-free method. This algorithm was pioneered by John Holland in the late 1960s. This method falls under a general class of algorithms which model evolutionary processes for use in optimization. For a good reference on evolutionary computing refer to [14]. A genetic algorithm is an attempt to numerically model Darwin's theory of evolution, based on natural selection, for optimization purposes. Unless otherwise specified, the algorithm begins by selecting a random initial population from the pool of candidate designs. The designs are represented as a vector of real numbers, for this example, a binary vector. The designs are evaluated and several actions may occur depending on the implementation of the algorithm. For example, a fraction of the best designs may be passed on to the next generation. These are sometimes referred to as the elite members of the population. Another action that may take place is called a mutation, where one or more bits in the vector representation may be changed before being passed on to the next generation:

$$\mathbf{x}_1 = [1 \ 1 \ 1 \ 1 \ 1 \ 1] \rightarrow \mathbf{x}_2 = [1 \ 1 \ 1 \ 1 \ 1 \ 0]$$

Figure 3.5: Mutation operation.

Some designs may experience a crossover, where two designs may trade portions of the design vector before being passed on to the next generation.

$$\begin{array}{l} \mathbf{x}_1 = [1 \ 1 \ 1 \ | \ 1 \ 1 \ 1] \\ \mathbf{x}_2 = [1 \ 0 \ 1 \ | \ 0 \ 1 \ 0] \end{array} \rightarrow \mathbf{x}_3 = [1 \ 1 \ 1 \ | \ 0 \ 1 \ 0] \quad (3.2)$$

Figure 3.6: Crossover operation.

Candidates that are not passed on to the next generation are dropped and new designs are sampled from the design space using these aforementioned operations. It is likely that through the generations these operations will result in designs that are near the current best designs. However, some designs may perform poorly as a result of the crossover and mutation operations. These designs that are far away from the current best design may

increase the likelihood of finding a better local optimum. As this procedure is repeated, only the best performing designs will be passed on to the final generation. Fig. 3.7 demonstrates how a genetic algorithm may search the design space of a 1-dimensional problem using a population size of 4.

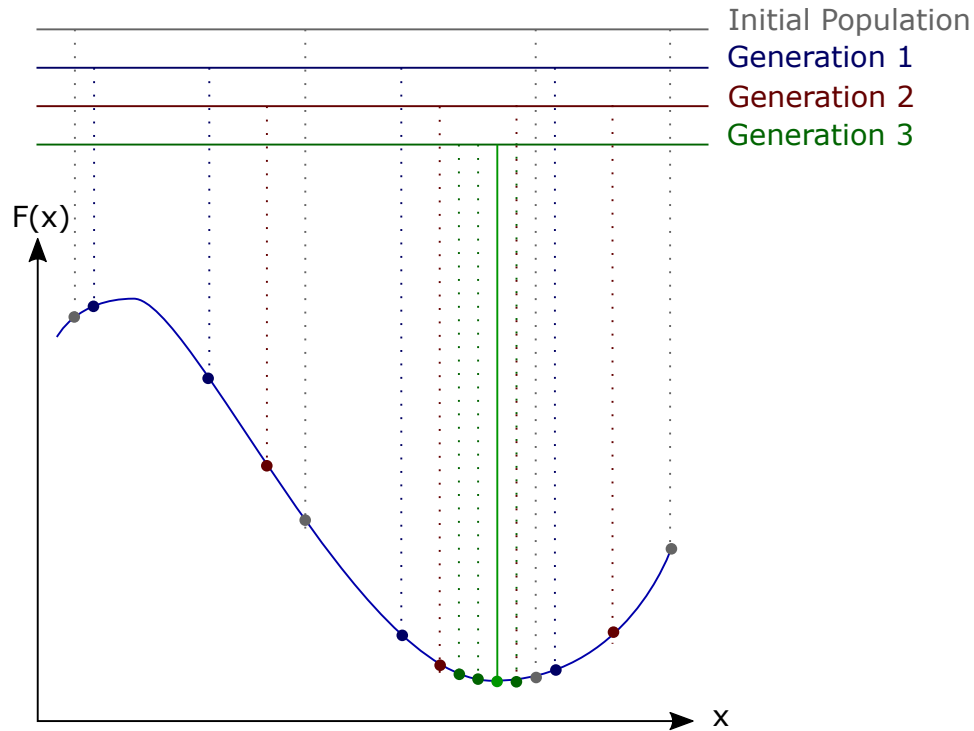


Figure 3.7: Genetic algorithm example.

The genetic algorithm would ideally cover most of the design space during an optimization and converge to a global optimum based on the spread of designs in the final generation. Global coverage however, is not guaranteed and local convergence requires many function evaluation. It is also important to note this described convergence will require many generations to be achieved. This has been grossly simplified in the example for demonstration purposes.

Hybrid-Strategies

Hybrid algorithms combine qualities of both gradient-free and gradient-based methods. This is achieved by executing both of algorithms together in search of a global optimum. In this work, a sequential hybrid approach is used.

Namely, a genetic algorithm method is used to search the domain and provide a good starting point for a gradient based or direct search method. The optimizer is then switched to an algorithm with better convergence properties to fine tune the optimization. This is demonstrated in Fig. 3.8.

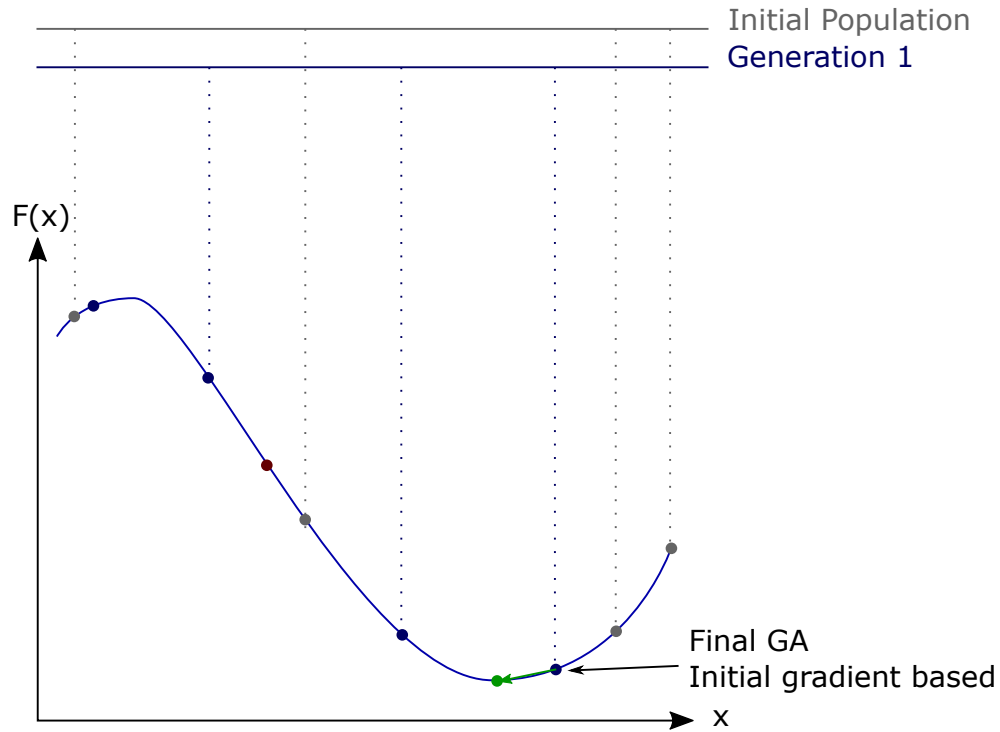


Figure 3.8: Sequential hybrid algorithm example.

In this example, a genetic algorithm is used to scan the design space for 2 generations before switching to a gradient-based method for rapid convergence. This method is usually slower than gradient based methods, but improves the probability of finding a global optimum. As an alternative, a memetic strategy could be used where a gradient based algorithm is nested in a genetic algorithm to improve the fitness of each individual in each generation. For the topology optimization problem investigated here, this implementation is not computationally tractable, and hence the sequential approach is used.

CHAPTER 4

TOPOLOGY OPTIMIZATION

In this work various applications of topology optimization are explored. Optimization of design topology aims to improve performance by making topological changes. Two designs have different topologies is a homeomorphism¹ does not exist between them. A topological design change is one that creates a new design that is not homeomorphic with the original design. To clarify the definition of topology optimization, consider the seven-bar truss in Fig. 4.1. The optimization will be described in the truss domain as it has clear visual resemblance to a graph.

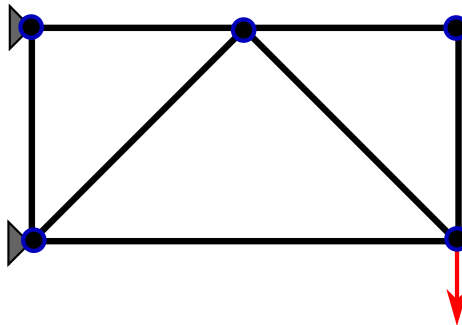


Figure 4.1: Truss ground structure

One intuitive strategy for defining a topology optimization problem is to define a ground structure, i.e., a description of all the possible elements and connections that may exist in a system, and then to choose a subset of these available elements as the design. Adding or removing an element from a design changes its topology. The graph corresponding to the truss in Fig. 4.1 may be used as a ground structure. Elements may be removed

¹A homeomorphism is a transformation that does not involve dividing or combining spaces.

and still produce a structurally stable (feasible) truss. Two of the nodes are fixed on the left side, and a load is applied at the bottom right node. A typical structural engineering problem formulation consists of minimizing the weight of the truss subject to stress and/or displacement constraints. When optimizing a truss structure such as this, there are three general types of design changes: topology, size, geometry (or shape).

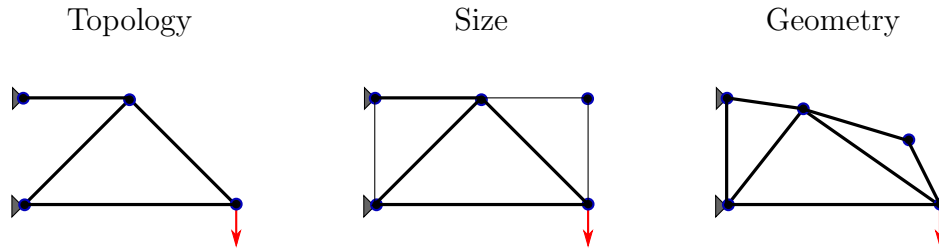


Figure 4.2: Optimal truss designs through various optimization.

When performing topology optimization we seek to determine which truss elements should exist and how they should be connected to minimize weight while satisfying the constraints [15–17]. Decision of member (edge) existence from the ground structure is a discrete problem. Size design optimization here refers to adjusting the cross sectional area of each existing truss bar, and can help further reduce weight while satisfying constraints [18–22]. Size optimization is generally a continuous problem, however, it may be treated as a discrete problem when constrained to a set of standard bar sizes. During optimization, members that do not bear load tend toward zero area. When these are removed, optimal designs may converge to optimal topologies (but not always). The final consideration is geometry or shape optimization, which considers the position of the nodes as design variables [23–28]. This is usually treated as a continuous optimization problem. In more complicated structural design problems shape optimization may involve distributed geometry. When considering all three optimization types in a single problem, significant increases in structural performance can be achieved.

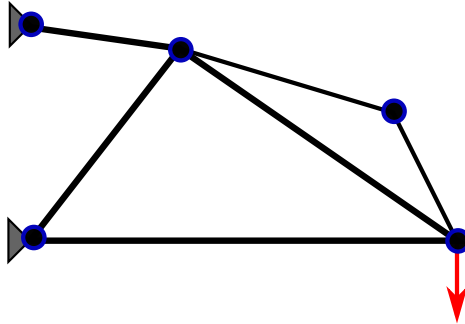


Figure 4.3: Truss with optimal topology, geometry, and size.

This combined problem, however, is challenging to optimize due to the need to optimize continuous and discrete variables together. A challenge associated with this is that the set of continuous variables changes when topology is changed. For this reason, researchers have developed an alternative relaxed parameterization of a truss which is based on ground structures. This next set of methods are categorized as the homogenization approach, developed by Bendsoe and Kikuchi [29].

4.1 Homogenization Method Formulation

This approach to structural optimization considers a continuous and bounded design domain. A designer must now decide how to distribute material on this domain.



Figure 4.4: Homogenization method design problem.

To do so, the domain is discretized into finite elements and whether material exists in each cell defines the material distribution. A binary value (0 or 1) is assigned to each element to specify material existence. Elements defined by 0 material are considered void, and elements defined by 1 are considered solid. This parameterization of the domain results in a large-scale problem with a set of binary design variables. Evaluating the structural behavior of a design using finite element analysis requires that each element has a small amount of material in it (not zero). Otherwise, the corresponding stiffness matrix would not be well-conditioned. Due to the discrete binary design representation, optimization based on this formulation requires use of zero-order (gradient-free) methods without the benefit of special problem structure to support efficient computation. This makes converging on a meaningful design difficult for anything but very small problems. Practical problems normally involve at least thousands of design variables.

A significantly more efficient strategy was discovered in the 1980s [29]. A continuous relaxation of the material distribution variables supports gradient-based methods. Each element is assigned a continuous value between 0 and 1 that indicates material density in that element. This strategy enables efficient solution of large problems due to special problem structure properties. This relaxation, however, poses an issue since an element with partial density is not always manufacturable. To create elements with partial density, some researchers have looked into micro-structure design [30], where the material properties of the intermediate density cells can be matched to a micro-structure. Gradient materials are also realizable using advanced manufacturing methods. If the engineering application for which topology optimization is used cannot support the inclusion of custom micro-structures or partial density elements, a penalization approach may be implemented to bias element density values toward void or fully-dense.

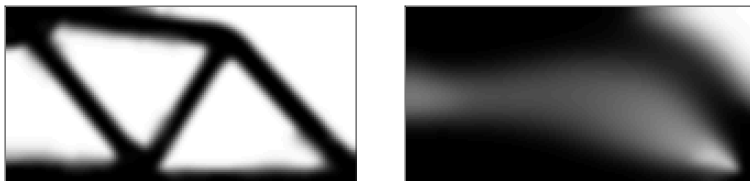


Figure 4.5: Penalized and un-penalized topology.

This avoids intermediate density elements, and is closer to a design that is

manufacturable using more conventional techniques. Consider the following equation:

$$C(x) = \alpha(x)C_0, \quad (4.1)$$

where the effective elasticity tensor, $C(\cdot)$, is the elasticity tensor, C_0 , penalized by a function, $\alpha(\cdot)$, in terms of the design variable, $0 \leq x \leq 1$. The penalty function can be designed to penalize intermediate material density values. Two common penalty functions used are an exponential function—used with the well-established method referred to as Solid Isotropic Material with Penalization (SIMP) [31] — and interpolation strategies, such as the Rational Approximation of Material Properties (RAMP) [32] method. Representative penalty functions are defined in Table 4.1.

Table 4.1: Penalization functions.

| | |
|-------------------|----------------------------------|
| $\alpha(x) = x^p$ | $\alpha(x) = \frac{x}{1+q(1-x)}$ |
| SIMP | RAMP |

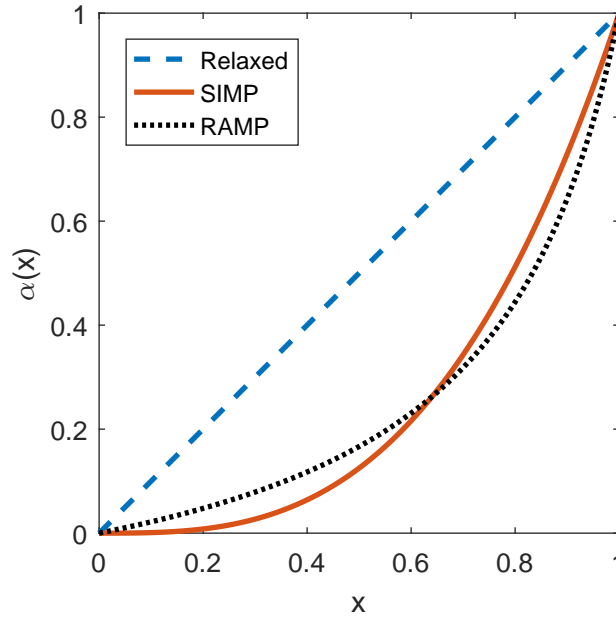


Figure 4.6: Penalization functions.

From the plot it is easy to see that elements with partially defined material retain a small amount density (material stiffness) when multiplied by a near-zero penalty. This makes the optimization favor elements with either fully defined solid material, or void. Generally, the RAMP formulation is preferred

over the SIMP formulation since the derivative of $\alpha(0) \neq 0$. However, this does not prevent the SIMP formulation from converging. Alternative formulations to these may be investigated to improve the convergence properties for specific applications.

4.2 Gradient Calculation

With this relaxed implementation, the optimization problem becomes differentiable. However, obtaining gradient information efficiently for the finite element model is difficult. To overcome this, the adjoint method for calculating gradients is used. The optimization problem objective function, represented here as $\Theta(\mathbf{x})$, may be expressed in terms of both independent and dependent variables solved for in a finite element program. To obtain the gradient of the objective with respect to a given design variable, the chain rule must be used. To help illustrate the use of the chain rule here, consider the following representation of the objective function:

$$\Theta(\mathbf{x}) = \sum \Pi(\mathbf{U}(\mathbf{x}), \mathbf{P}(\mathbf{x}), \mathbf{x}), \quad (4.2)$$

where the objective is the summation of a function, $\Pi(\cdot)$, that depends on displacement field, $\mathbf{U}(\mathbf{x})$, load field, $\mathbf{P}(\mathbf{x})$, and the vector of the design variables, \mathbf{x} . Using this representation, it becomes clear where dependencies on the design variable are possible. To obtain the gradient of this function, the chain rule can be used:

$$d\Theta(\mathbf{x}) = \frac{\partial \Pi}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{x}} \quad (4.3)$$

From this equation, it is clear that using a finite difference strategy for obtaining the gradient is expensive since calculating the sensitivity of the displacement field with respect to each design variable requires multiple finite element analyses. To avoid this, the adjoint method for calculating the gradient can be used. The residual of finite element analysis, which should be approximately zero, is differentiated and added to the gradient of the objective function.

$$d\Theta(\mathbf{x}) = \frac{\partial \Pi}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U} + \mathbf{K} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} - \frac{\partial \mathbf{P}}{\partial \mathbf{x}} \right]. \quad (4.4)$$

Where, \mathbf{K} , represents the stiffness matrix, and an adjoint (dual) variable, $\boldsymbol{\lambda}$, multiplies the differentiated residual equation. This is a fair mathematical operation since the residual term is zero. Rearranging this equation, the following representation can be obtained:

$$d\Theta(\mathbf{x}) = \left[\frac{\partial \Pi}{\partial \mathbf{U}} + \boldsymbol{\lambda}^T \mathbf{K} \right] \frac{\partial \mathbf{U}}{\partial \mathbf{x}} + \left[\frac{\partial \Pi}{\partial \mathbf{P}} - \boldsymbol{\lambda}^T \right] \frac{\partial \mathbf{P}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U} \right]. \quad (4.5)$$

At this point, there are many unknowns in the system. However, by inspection, $\boldsymbol{\lambda}$ may be chosen in such a way as to eliminate the expensive partial derivative terms. This will become clear when considering problem specific loading and boundary conditions. Since the procedure for obtaining the gradient at this point becomes problem dependent, the derivation of the specific sensitivity calculation is left for future sections.

4.3 Compliance Optimization

The success of the homogenization method is due to the flexibility of this method to handle problems of various physics without major modification to the design optimization method. Topology optimization methods have been under development for a few decades and are now beginning appear in many commercial finite element packages. However, many uses of the topology optimization method are centered around a particular class of objective function: compliance. A typical topology optimization problem for the truss problem is presented in the following equation:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & C = \int UPd\Omega \\ \text{s.t.} \quad & V \leq V_p \\ & R(x) \geq R_{\min}, \end{aligned} \quad (4.6)$$

where the compliance of the structure, C , is minimized with respect to material density, x . This is subject to a volume constraint, V_p , and minimum radius constraint, R_{\min} . To obtain the gradient information for the compli-

ance problem, Eq (4.5) can be reduced. It is restated here:

$$d\Theta(\mathbf{x}) = \left[\frac{\partial \Pi}{\partial \mathbf{U}} + \boldsymbol{\lambda}^T \mathbf{K} \right] \frac{\partial \mathbf{U}}{\partial \mathbf{x}} + \left[\frac{\partial \Pi}{\partial \mathbf{P}} - \boldsymbol{\lambda}^T \right] \frac{\partial \mathbf{P}}{\partial \mathbf{x}} + \frac{\partial \Pi}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U} \right].$$

Using problem structure, such as a zero Dirichlet boundary, allows for the reduction of this equation. Consider splitting variables defined by free and prescribed degrees of freedom in the residual equation:

$$\begin{bmatrix} \mathbf{K}^{ff} & \mathbf{K}^{fp} \\ \mathbf{K}^{pf} & \mathbf{K}^{pp} \end{bmatrix} \begin{bmatrix} \mathbf{U}^f \\ \mathbf{U}^p \end{bmatrix} = \begin{bmatrix} \mathbf{P}^f \\ \mathbf{P}^p \end{bmatrix}, \quad (4.7)$$

where the boundary simplifications can be used to reduce the gradient equation:

$$d\Theta(\mathbf{x}) = \left[\frac{\partial \Pi}{\partial \mathbf{U}^f} + \boldsymbol{\lambda}_f^T \mathbf{K}^{ff} + \boldsymbol{\lambda}_p^T \mathbf{K}^{pf} \right] \frac{\partial \mathbf{U}^f}{\partial \mathbf{x}} - \boldsymbol{\lambda}_p^T \frac{\partial \mathbf{P}^p}{\partial \mathbf{x}} + \boldsymbol{\lambda}_f^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U}^f \right]. \quad (4.8)$$

The derivative displacement terms on the fixed boundary drop out, since the displacement is zero. The derivative of the loading terms on the free domain also drop out, since the loading is constant. The derivative of the compliance with respect to the design variable also drops out, since compliance is not explicitly in terms of the design variable. With the gradient equation in this form, solving for the adjoint variable becomes clear. If $\boldsymbol{\lambda}_p$ is chosen to be zero, the equation reduces to the following form,

$$d\Theta(\mathbf{x}) = \left[\frac{\partial \Pi}{\partial \mathbf{U}^f} + \boldsymbol{\lambda}_f^T \mathbf{K}^{ff} \right] \frac{\partial \mathbf{U}^f}{\partial \mathbf{x}} + \boldsymbol{\lambda}_f^T \left[\frac{\partial \mathbf{K}^{ff}}{\partial \mathbf{x}} \mathbf{U}^f \right], \quad (4.9)$$

where $\boldsymbol{\lambda}_f$ can be obtained as the solution to a linear system of equations to eliminate the a partial derivative term.

$$\boldsymbol{\lambda}_f^T \mathbf{K}^{ff} = -\frac{\partial \Pi}{\partial \mathbf{U}^f} = \mathbf{P}^f = \mathbf{K}^{ff} \mathbf{U}^f \quad (4.10)$$

In fact, the linear system of equations does not need to be solved numerically, analytically solving this equation reveals $\boldsymbol{\lambda}_f^T = \mathbf{U}^f$. The gradient equation is then reduced to the following form,

$$d\Theta(\mathbf{x}) = \boldsymbol{\lambda}_f^T \left[\frac{\partial \mathbf{K}^{ff}}{\partial \mathbf{x}} \mathbf{U}^f \right] = \mathbf{U}^f \left[\frac{\partial \mathbf{K}^{ff}}{\partial \mathbf{x}} \mathbf{U}^f \right] \quad (4.11)$$

This expression allows for the calculation of the gradient using known quantities obtained from the finite element analysis, hence further reducing the computational expense. This makes the compliance objective desirable for fast optimization.

4.4 Filtering Methods

To handle the satisfaction of the minimum radius constraint, researchers have implemented image processing techniques. Namely, introducing a filter to bias the design towards larger or smaller void areas. These filters are applied on localized neighborhoods to enforce minimum radius constraints and define how material is updated between iterations. Introducing this filter effectively removes the constraint from the optimization problem, since it becomes inherently satisfied after several iterations. Optimal designs obtained by various filtering techniques are presented in Fig. 4.7.

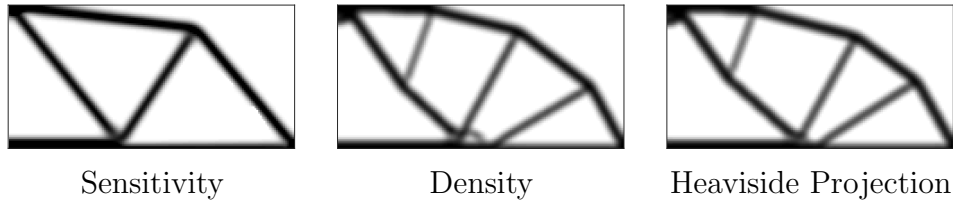


Figure 4.7: Optimal topology by filtering method.

In sensitivity filtering, the filter is applied solely to the gradient of the objective function to redistribute material on the domain [33]. In density filtering, the filter is applied both to the gradient of the objective and to the volume constraint to redistribute material on the domain [34]. The Heaviside projection filter uses the Heaviside function in an inner optimization loop to redistribute material on the domain [35]. Changing a filter will likely change the final result of the optimization [36]. When designing a structure, it is a good idea to obtain an array of designs by optimizing with various filters. After post-processing these designs, commercial finite element analysis (FEA) software can be used to compare fairly the performance of each design.

4.5 Optimization Routines

To solve the presented homogenization type problem, many researchers are designing optimization algorithms to improve computational efficiency. These may be general algorithms, or tailored to perform well for one specific problem formulation. Consider once more the truss design problem discussed in this chapter. The similar optimization problem is solved here using the homogenization approach with three different optimization algorithms.

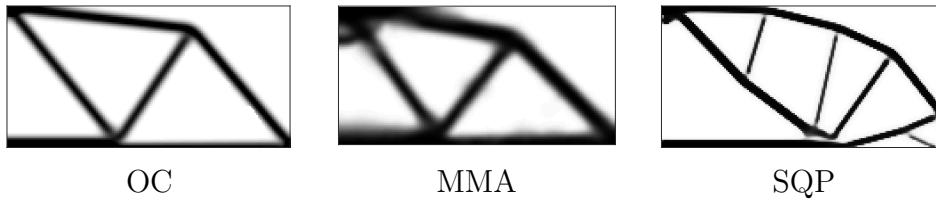


Figure 4.8: Optimal topology by solver.

The Optimality Criteria (OC) method was designed specifically to optimize a problem with a single equality constraint. This is a fair modification since the optimization solution is generally on the constraint boundary. Under this assumption, the algorithm can converge quickly to a solution. The Method of Moving Asymptotes (MMA) algorithm was designed to solve large-scale optimization problems and can handle multiple constraints. In the topology optimization community, this is the algorithm of choice. The final solution was obtained using MATLAB's `fmincon()` sequential quadratic programming (SQP) algorithm. It is interesting to see that each algorithm converges on a somewhat different design, even when sharing the same analysis and filter.

4.6 Homogenization Approach as a Generative Algorithm

The homogenization approach is enabled by clever problem formulation and optimization algorithms. It is designed to work as an optimization method and is considered as such. When analyzing the methodology from a generative perspective, new insights emerge. Since the homogenization approach uses a finite element mesh for design, it can be seen as a large scale graph, where each element of the mesh is a node on the graph. The density of

material in an element can be considered the label of a node. Topology optimization methods then used localized rules to enforce minimum radius constraints, similar to cellular automaton which uses localized rules to update cells. When recognizing this, the entire problem manifests itself as an application of cellular automaton with discrete cells and continuous design rules (gradient based update). From this point of view, this makes homogenization topology optimization methods the most successful and widespread use of cellular automaton in engineering design. This observation is briefly discussed in a recent review [37].

CHAPTER 5

GENERATIVE ALGORITHMS FOR HEAT SPREADER DESIGN

In this chapter the generative design methodology will be applied to passive heat spreader¹ design. The procedure for implementing this method is presented in the following figure.

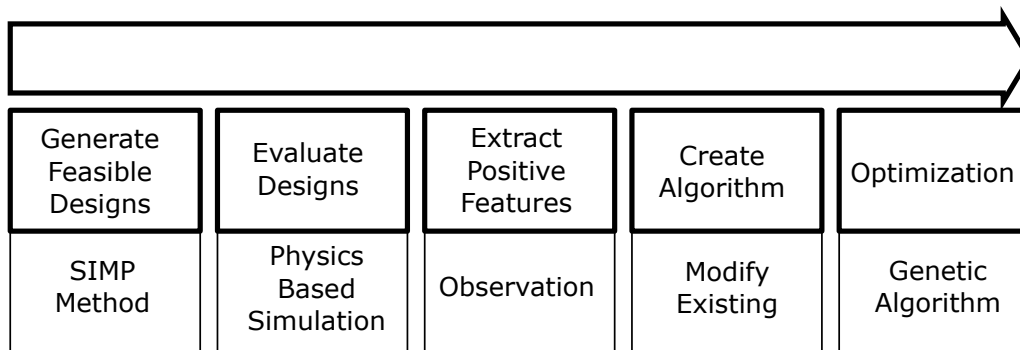


Figure 5.1: Generative design methodology for heat spreader design.

As an early implementation of this methodology, many of these steps are done manually. First a literature review of topology optimization for heat transfer is conducted. Observations are made to identify characteristics of optimal structures presented in the literature. Then a generative algorithm is selected that is capable of producing designs with similar features. This generative algorithm is then used in optimization as a design abstraction to support efficient search for improved designs. It is my vision to automate this process in the future to create an autonomous, generalizable, design tool to aid engineers. Such automation is especially important for cases where manual observation of important design features is impractical, or when existing available generative algorithms cannot produce the desired features. This chapter will walk through the generative design procedure in detail for the passive heat spreader design study.

¹A heat spreader is used to direct heat from a heat source to a heat sink.

5.1 Heat Spreader Design Problem

To best evaluate the generative design method (GDM) for designing heat extraction topology, a benchmark problem is used. Consider a homogeneously heated design domain, as shown in Fig. 5.2.

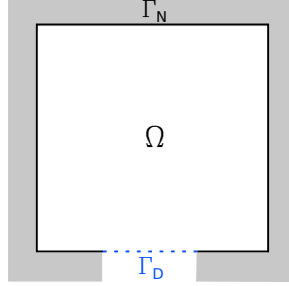


Figure 5.2: Homogeneously heated design domain.

The design domain, Ω , is bounded by the solid black line. The temperature is fixed at zero on the Dirichlet boundary, Γ_D , represented by the dashed line. The Neumann boundary, Γ_N , is adiabatic and restricts heat flux out of the domain. The steady-state conductive heat transfer across the domain can be represented by the following governing equations:

$$\nabla \cdot (\kappa \nabla T) + q = 0 \text{ on } \Omega \quad (5.1)$$

$$T = 0 \text{ on } \Gamma_D \quad (5.2)$$

$$(\kappa \nabla T) \cdot n = 0 \text{ on } \Gamma_N, \quad (5.3)$$

where T is the spatially-varying temperature state variable, q is the heat generated, and κ is the thermal conductivity of the material in the domain, Ω . The thermal compliance of the system is commonly used as a heat transfer optimization objective function:

$$C = \int \nabla T \cdot \mathbf{q} \, dA = \int \nabla T (k \nabla T) \, dA. \quad (5.4)$$

This measure sums the element-wise displacement and loading vectors (temperature and heat flux) at each finite element node.

$$C = U'P \quad (5.5)$$

The design optimization problem considered here is:

$$\underset{\mathbf{x}}{\text{minimize}} \quad C(\mathbf{x}) \quad (5.6)$$

$$\text{subject to} \quad V(\mathbf{x}) \leq V_p \quad (5.7)$$

$$R(\mathbf{x}) \geq R_p \quad (5.8)$$

where the goal of the optimization is to minimize the thermal compliance of the system. The amount of material, $V(\mathbf{x})$, is constrained to be less than a prescribed value V_p . The radius of any conductive path elements, $R(\mathbf{x})$, must be larger than a prescribed value R_p . The design vector, \mathbf{x} , is the relaxed design representation that maps to element material density. The specific meaning of elements in the design vector will change between the different design methods used.

5.2 Heat Spreader Topology Optimization Review

Heat spreader design using topology optimization has been investigated for numerous applications. Initial work in this domain was conducted by Hansen et al. [38], where the finite volume method was used with topology optimization to solve a planar heat conduction problem. The optimal structure obtained resembled a branching ‘five-finger’ structure. A similar structure is observed when optimizing a multi-objective problem when heat conduction is dominant [39]. Beyond the SIMP parameterization of topology optimization, researchers have also investigated the use of level set methods, where similar branching patterns emerged as optimal [40]. When scaling to three dimensions, it was observed that tree-like structures were again found to be ubiquitous features in optimal designs [41–43]. These structures, however, are not limited to heat conduction exclusively. When considering heat convection, researchers observed similar dendritic patterns, [44–47]. This type of structure seems to correlate more with the governing equations of the physics than the loading conditions. However, there is still a correlation with loading conditions. For example, the designs change somewhat when heat convection becomes the dominant mode of heat transfer. Yet, the topology optimization results are still dendritic in nature, and it can be concluded that dendritic

patterns represent a class of optimal features for conductive and convective heat transfer. In this work, heat conduction is considered exclusively.

5.3 SIMP Optimization

To validate GDM solutions, the SIMP optimization will be performed here to confirm the dendritic structure for this specific optimization. Recall that the gradient calculation for compliance results in the following equation for a particular element:

$$d\Theta(x) = \mathbf{U}^f \left[\frac{\partial \mathbf{K}}{\partial x} \mathbf{U}^f \right]. \quad (5.9)$$

This avoids the need to solve a linear system to obtain the adjoint variable, and therefore makes the compliance metric favorable for efficient computation. The topology optimization problem was solved for three mesh densities using the MMA algorithm. Fig. 5.3 illustrates the results from each of these uniform meshes, with increasing refinement from left to right.

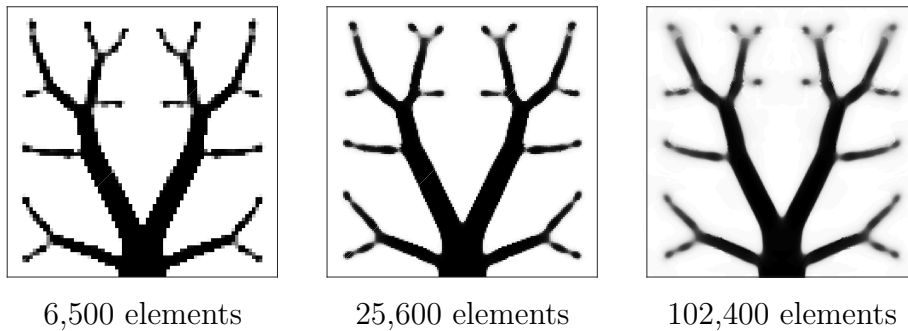


Figure 5.3: SIMP solutions for heat spreader design.

From this figure it is clear the same dendritic structure results for all mesh densities. This confirms the results presented in literature for passive heat spreaders dominated by heat conduction and motivates the investigation of this particular structure. The results obtained using the SIMP method will be used to benchmark the performance of the generative method.

5.4 Generative Algorithm Selection

To find an algorithm to produce dendritic structures, attention is first directed to existing applications in engineering design. As an alternative to SIMP topology optimization, Constructal Theory [48] has been used to design structures for heat transfer. Pioneered by Adrejan Bejan, these approaches are inspired by biological systems. The optimal topology for heat conduction problems resemble the naturally occurring phenomena of venation, the arrangement of veins in a leaf. In heat transfer applications, some researchers have recognized the tendency of optimal heat transfer topology to resemble dendritic patterns and capitalize on this to perform optimization studies. Bejan et al. [48] use an approach similar to Lindenmayer-systems to develop tree-like structures to solve heat transfer problems. Salakij et al. [49] and Heyman et al. [50] solve the topological design for convective heat transfer using an algorithm that generates fractal like patterns directly.

However, the search for a generative algorithm should not be restricted to existing use in engineering design. To perform a targeted search of the design domain, a generative algorithm that efficiently produces complex dendritic structures must be identified. For this reason, a broader search of algorithms is conducted outside of engineering design. Researchers in computer graphics have worked to efficiently and accurately reproduce leaf structures. Meinhardt produced dendritic structures using reaction diffusion models [51]. Rodkaew et al. developed an algorithm based on particle systems which grew dendritic structures to an origin point [52]. Runions et al. developed a similar algorithm for the purpose of visualization based on particle systems, which grew dendritic structures away from an origin point [7, 53]. It is interesting to note that all the algorithms described in Chapter 2 have been used to create dendritic structures.

A strategy has been created for this study to assess these algorithms, and it is presented in Table 5.1. The algorithms have been regrouped based on common themes. Namely, grammar-based algorithms, interaction-based algorithms, and physics-based algorithms. These categorizations align well with observations made in Chapter 2. The algorithms are evaluated based on the following criteria: number of design variables, whether branches overlap, and whether the design lies inherently within the design domain. These three metrics capture the utility of a particular algorithm in optimization

of a specific application. A preferred algorithm would have a small number of design variables, satisfy all constraints, and produce the desired class of structure. Since all of the algorithm produce dendritic patterns, they score evenly in last criteria. The remaining metrics are presented in the table.

Table 5.1: Generative algorithm assessment.

| Basis | Generative Algorithm | Design Vars | Overlap | Boundary. Conds. |
|-------------|-------------------------------|-------------|---------|------------------|
| Grammar | L-System (LindenMayer) | Med | Yes | No |
| | Constructal Theory (Bejan) | Med | Yes | No |
| Interaction | Reaction Diffusion(Meinhardt) | Low | No | Yes |
| | Particle System (Rodkaew) | Low | No | Yes |
| | Space Colonization (Runions) | Low | No | Yes |
| Physics | Erosion Model (Bejan) | High | No | Yes |
| | SIMP (Sigmund) | High | No | Yes |

From the table, it is clear that the algorithms within a category score identically. With this in mind, it is not as important to identify one algorithm to use, but to identify what type of algorithm to use. For this optimization problem, it is important to prevent member overlap and remain within the design domain. Two types of algorithms satisfy these conditions (interaction and physics). However, one of them requires a small number of design variables, while the other requires a large number of design variables. For this reason, an ‘interaction’ based algorithm is investigated. For use in optimization, the space colonization algorithm was chosen since the algorithm growth procedure transfers intuitively to heat transfer applications. This observation will be described in more detail in the following section.

5.4.1 Space Colonization Algorithm

The space colonization algorithm was designed to efficiently produce realistic leaf-like structures. It does so by following rules of the *canalization hypothesis* [8], which suggests that leaf veins grow towards hormone centers. These hormone centers are called auxins and are scattered throughout the leaf. The algorithm begins from a source node or initial stem, shown in green Fig. 5.4. The nearest vein node to an auxin proceeds to grow towards that auxin by a fixed step. If a vein node is equidistant from two auxins, it grows in the

average direction of those auxins by a fixed step. The algorithm iterates until all auxins are reached by a vein node. At this point the topology can be post processed to produce realistic leaf veins. In this thesis, the thickness of each vein node will increase linearly from the extremities towards the origin node. Alternate nonlinear thickness distributions may be defined.

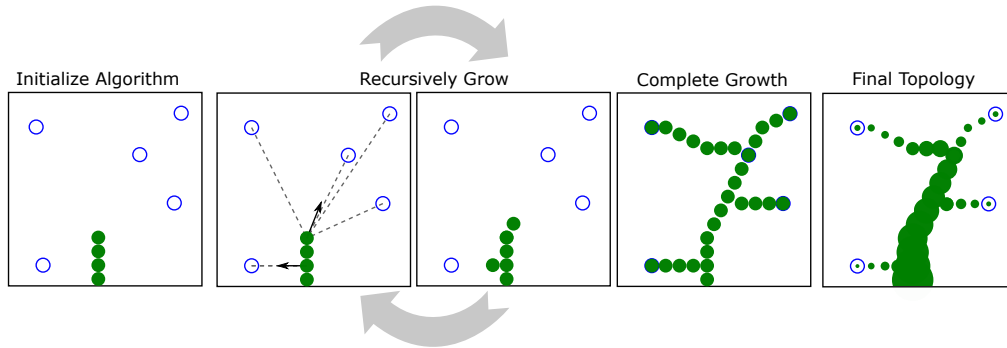


Figure 5.4: Space colonization algorithm.

The procedure in which the space colonization algorithm grows branches towards auxins shows promise for heat transfer, where cooling channels would grow towards heat sources. This method to produce dendritic structures has also been tested in three dimensions [53]. As a demonstration, the space colonization algorithm was used to produce the 3D structure presented in Fig. 5.5.

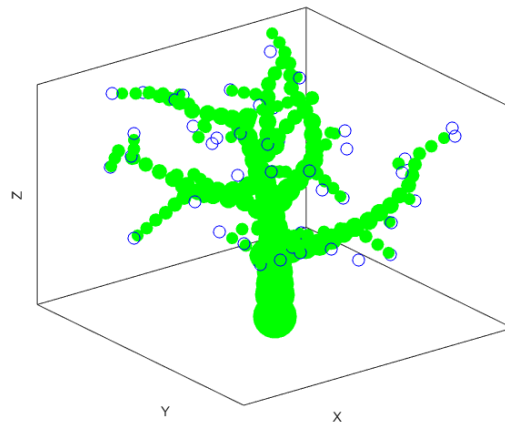


Figure 5.5: Space colonization algorithm 3D example.

Using this algorithm, a variety of unique dendritic structures can be generated by changing the algorithm growth parameters. These include, but are not limited to:

- Number of auxins
- Auxin location
- Intervals at which auxins are introduced
- Thickness at each node

For the subsequent studies, the algorithm will be adjusted to satisfy the constraints of the optimization problem. Namely, vertical symmetry will be enforced for topologies and a total of eight auxins are used for optimization. To evaluate a topology obtained by the space colonization algorithm, the topology will be projected onto a regular mesh.

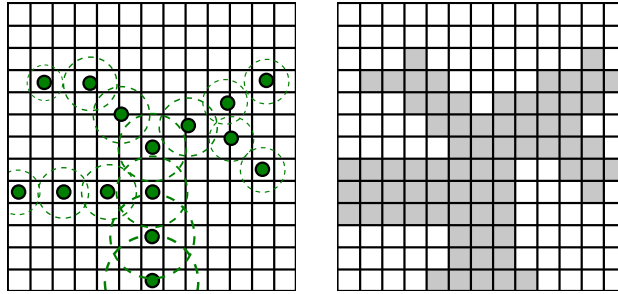


Figure 5.6: Projection of space colonization algorithm onto a regular mesh.

Specifically, the elements whose centroids lie within a radius of the vein node will be assigned thermally conductive material properties. Using a regular mesh to define the space colonization topology enables the use of the same finite element program to evaluate both the SIMP result and the generative algorithm result. An important potential advantage of the GDM is the ability to use a spatially-varying mesh that could increase accuracy for a given number of elements.

5.5 Optimization Results

For this study, two optimization routines will be investigated. The first of which is a SIMP implementation using the MMA algorithm, the second implementation is the GDM using a genetic algorithm to find a good starting point for the SIMP algorithm in the hybrid approach. The space colonization algorithm will use 16 design variables to produce a dendritic topology. The

GA is set to evaluate 100 total designs with a population size of 10 before passing the best topology to the SIMP algorithm. The design domain is normalized to 1x1m space. There is a uniform heat flux of $20W/m^2$ applied on the domain. The maximum amount of material is restricted to 30% of the domain area. The minimum radius is constrained to be greater than $\frac{1}{20}$ of the design domain width. The results of these optimizations are presented in Fig. 5.7.

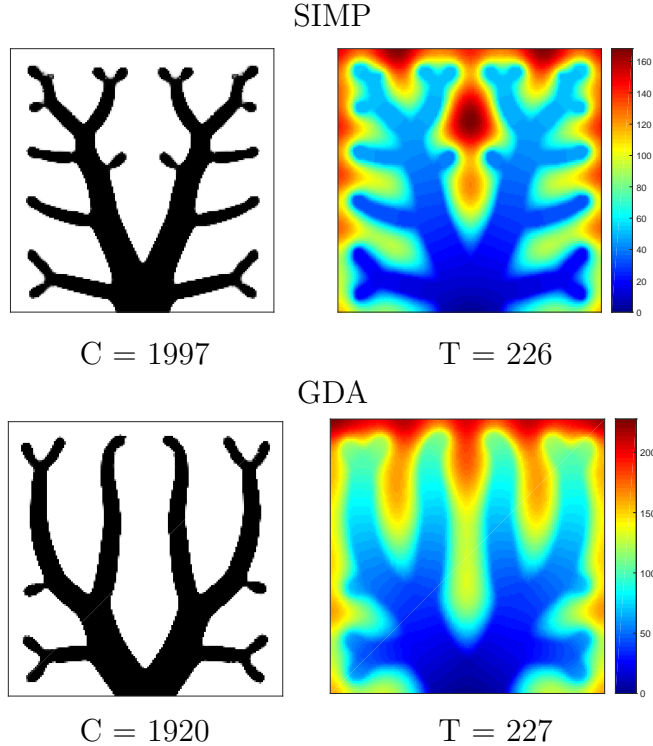


Figure 5.7: Compliance Optimization Results

Interestingly, the hybrid optimization presents a topology which is less dendritic and has a better compliance value. This raises a question regarding the optimality of highly dendritic structures. The maximum temperature on the domains are roughly equal, however, the distribution of high temperatures on the domain are somewhat different. The GDA solutions has generally less deep red on the domain, however, the SIMP solutions has more deep blue. Recognizing this different raises a question about the validity of the compliance metric, which clearly does not capture enough information about heat flow on the domain. Whether a single metric can capture this is an open research question. Additional metrics for topology optimization will be

investigated in the next chapter to answer this question. Prior to investigating alternative objectives, the hybrid optimization result must be analyzed further. Since a genetic algorithm is used in this optimization, there is no guarantee for convergence, or even to obtain the same solution twice. For example, the 5 topologies presented in the top row of Fig 5.8 were obtained using the GA in 5 different optimization trials, all other things being equal.

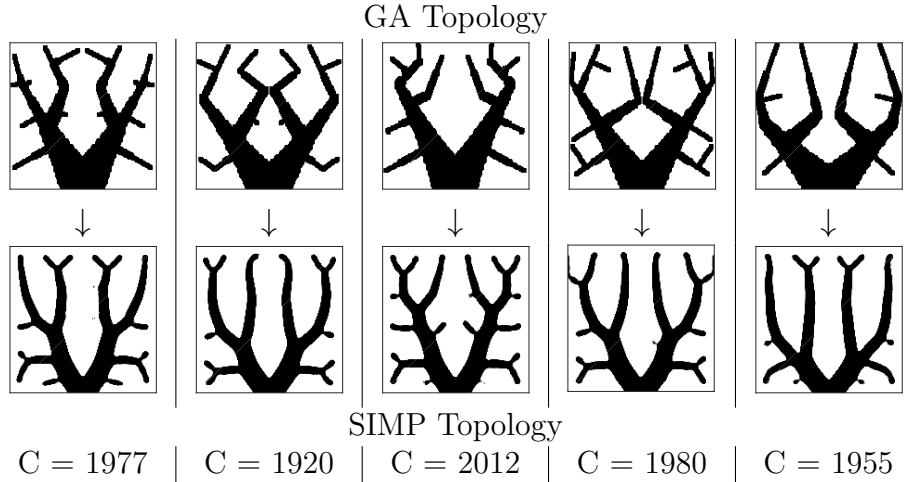


Figure 5.8: Sample solutions obtained using hybrid approach.

These five topologies were used to initialize the SIMP algorithm, which then generated the five topologies directly under them. It is clear from the image that none of the initial topologies remained unchanged by the SIMP algorithm. Yet, the structure of the SIMP solutions resemble their initial GA obtained structures. This highlights the non-convex nature of this design problem where many local minima exist. The final solutions vary in performance based on the initial starting point for the SIMP algorithm, namely, solutions with better starting objective values often converged to better optimized structures. Increasing the population size in the GA will likely result better starting topology for the SIMP algorithm, but increases the computational expense of the algorithm. A GA population size of was used to demonstrate that even a small number of GA evaluations can result in a significant impact on final performance. The next chapter will defend these claims with additional examples.

CHAPTER 6

ALTERNATIVE OPTIMIZATION FORMULATIONS

While optimizing thermal compliance results in an optimization with inexpensive gradient calculations, this objective is not completely aligned with the goal of a heat spreader, general temperature minimization. To design the most efficient heat spreader, several topology optimization formulations are investigated in this chapter. The design problem will mirror that presented in the previous chapter. The figure below illustrates the design domain, followed by the baseline problem formulation.

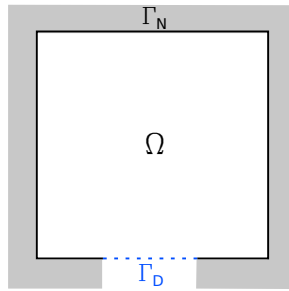


Figure 6.1: Homogeneously heated design domain.

$$\begin{aligned}
 & \underset{\mathbf{x}}{\text{minimize}} && \Theta \\
 & \text{subject to} && V(\mathbf{x}) \leq V_p \\
 & && R(\mathbf{x}) \geq R_p
 \end{aligned} \tag{6.1}$$

Using this formulation with a fixed zero temperature boundary results in the following element-wise gradient equation if λ_p is chosen to be zero,

$$d\Theta(\mathbf{x}) = \left[\frac{\partial \Pi}{\partial \mathbf{U}^f} + \boldsymbol{\lambda}_f^T \mathbf{K} \right] \frac{\partial \mathbf{U}^f}{\partial \mathbf{x}} + \boldsymbol{\lambda}_f^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U}^f \right]. \tag{6.2}$$

This general equation is a result of problem structure and will hold for

any of the objectives used here. The difference between the design problems manifests in the adjoint variable, which will be different depending on the objective function. The formulations presented here relate to temperature requirements in practical systems. For each problem formulation, the adjoint equation is defined and a topology optimization solution is presented. A discussion of these results is presented in Sec. 6.3, after all objective functions have been described.

6.1 Average Temperature Minimization

As a first step away from thermal compliance optimization, consider the following formulation minimizing the temperature sum on the domain.

$$\underset{\mathbf{x}}{\text{minimize}} \quad \Theta(x) = \int_{\Omega} T \quad (6.3)$$

$$\text{subject to} \quad V(\mathbf{x}) \leq V_p \quad (6.4)$$

$$R(\mathbf{x}) \geq R_p \quad (6.5)$$

The temperature sum is aligned with average temperature, which is simply the temperature sum scaled by the number of elements. To obtain the sensitivities for this objective function, the following adjoint equation must be solved:

$$\lambda_f^T \mathbf{K} = -\frac{\partial \Pi}{\partial \mathbf{U}^f} = -\mathbf{1}. \quad (6.6)$$

The resultant adjoint variable can be used to evaluate the sensitivity equation,

$$d\Theta(\mathbf{x}) = \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U}^f \right]. \quad (6.7)$$

When using the SIMP penalty method to optimize this problem formulation, the following structure is obtained, Fig 6.2.

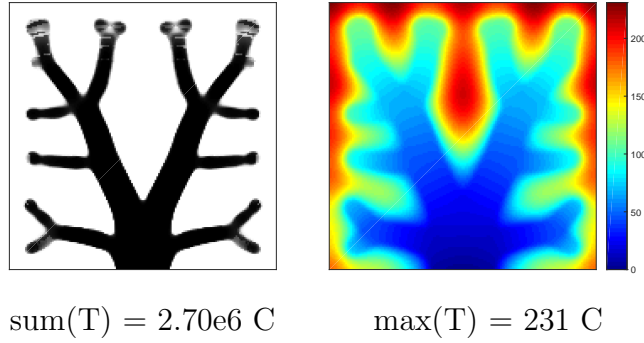


Figure 6.2: Temperature sum optimization result.

This result resembles the compliance solution obtained in the previous section, however, the maximum temperature on the domain increased. The algorithm also appears to have trouble satisfying the minimum radius constraint near the extremities of the branches. This highlights the numerical instabilities that exist in the algorithm. Using the GDM method, a strikingly different structure is obtained, Fig 6.3.

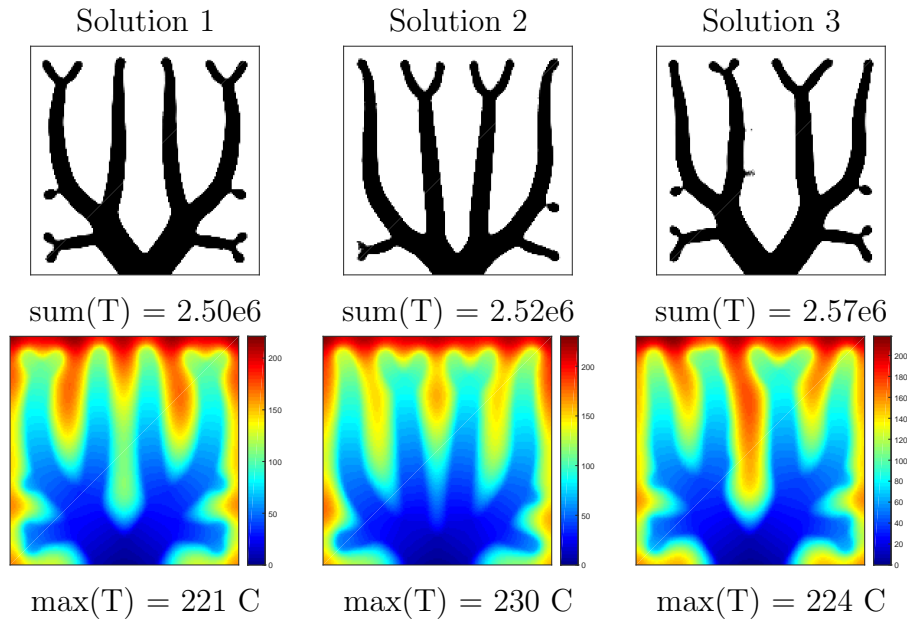


Figure 6.3: Maximum temperature optimization solutions obtained by hybrid approach.

These solutions are not highly dendritic when compared to the compliance solutions. The temperature sums decrease significantly with these designs dominated by long slender conductive paths. It is also interesting to note

the asymmetry that is present in some of these solutions. This asymmetry is likely to be due to the implicitly enforced minimum radius constraint, which is fairly large, and numerical instabilities. It shows that improved solutions can be obtained, though they may be suboptimal. However, the best design obtained is symmetric. The maximum domain temperatures of these solutions are not significantly lower than those obtained with a compliance objective, however, the temperatures on the domain look generally lower (there is less red on the domain). If what we really want to minimize is maximum temperature, this objective function is a proxy for the actual objective.

6.2 Max Temperature Minimization

A more direct representation of temperature optimization is the minimization of the maximum temperature on the domain. This particular implementation is challenging to solve since the max function is non-differentiable. To address this, a p-norm approximation for the maximum temperature is used.

$$T_{\max} \approx \|T\|_p = \left(\sum_{i=1}^n |T_i|^p \right)^{1/p} \quad (6.8)$$

As the norm exponential increases, this function more accurately represents the maximum temperature on the domain. However, the derivative becomes less well behaved. Hence, there is a tradeoff in selecting the appropriated p-norm for optimization. For this study, a p-norm value of 10 is used. To obtain the gradient for this objective the following adjoint equation must be solved:

$$\lambda_f^T \mathbf{K} = -\frac{\partial \Pi}{\partial \mathbf{U}^f} = -\frac{\partial \|T\|_p}{\partial T} = -\frac{T_i |T_i|^{p-2}}{\|\mathbf{T}_p^{p-1}\|}. \quad (6.9)$$

Since the boundary conditions did not change, the gradient equation has the same form:

$$d\Theta(\mathbf{x}) = \boldsymbol{\lambda}^T \left[\frac{\partial \mathbf{K}}{\partial \mathbf{x}} \mathbf{U}^f \right]. \quad (6.10)$$

Solving this maximum temperature approximation problem results in the following topology and heat map.

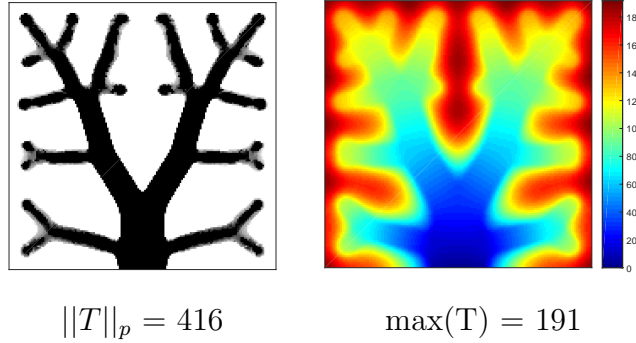


Figure 6.4: Max temperature optimization result.

For this objective, a very different type of dendritic structure is obtained. The branches appear to cover much of the surface of the design domain. The maximum temperature is greatly reduced when compared to the last two problem formulations. However, the p-norm approximation of the temperature has over 100% error. This does not hinder the results since the gradient is decreasing a maximum temperature directly. In other words, while the absolute value of the approximate objective function is inaccurate, it is well-aligned with the desired objective function. Enforcing a temperature constraint with the p-norm would be inappropriate, as constraint satisfaction would not be guaranteed. To improve these solution, a good starting topology will be obtained using the hybrid approach. Once more, a genetic algorithm is used to find a good starting topology using the space colonization algorithm abstraction. The resultant topology is passed to the MMA algorithm where the where the gradient based method is used to fine tune the solution.

What is particularly interesting is that there exists multiple different solutions that have the same objective value. Notice that solution 2 and 3 are asymmetric, this is typically a characteristic of suboptimal designs. With that in mind, these solutions are likely suboptimal, however, they are perform better than the previously obtained symmetric designs.

6.3 Summary of Results

In this chapter, an investigation of problem formulations for heat spreader design was conducted. Two alternative problem formulations for compliance were presented. While these proposed formulations are slightly more

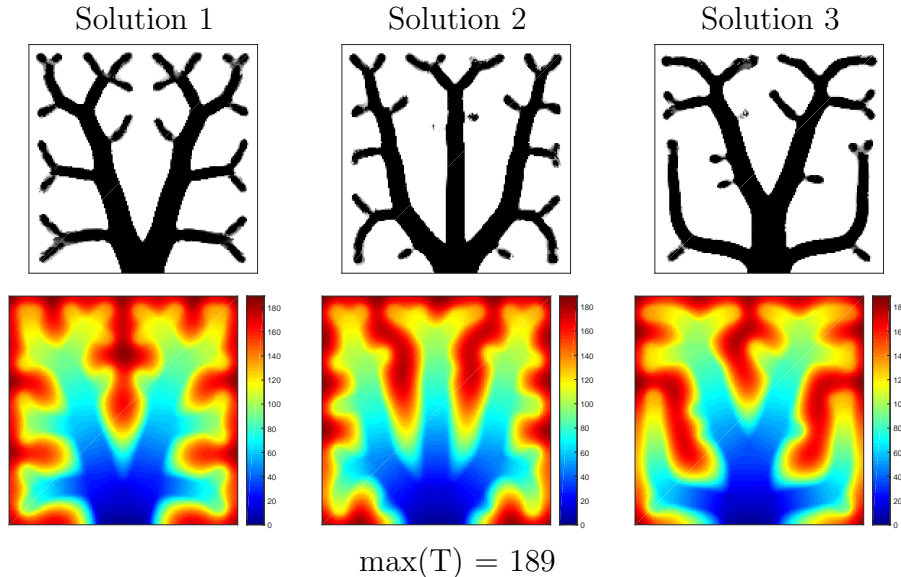


Figure 6.5: Maximum temperature minimization solutions obtained using the hybrid approach.

expensive computationally when calculating the adjoint variable, they offer much improvement in their results. Table 6.1 presents the computational expense of solving the SIMP topology optimization problem for each formulation. The expense is measured in number outer loop MMA iterations and the total time of the MMA optimization.

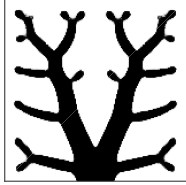
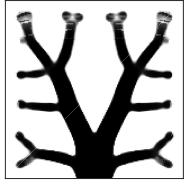
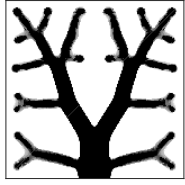
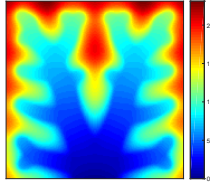
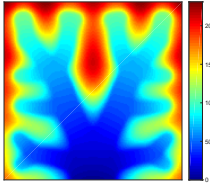
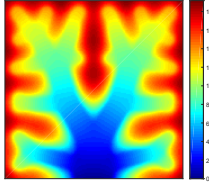
Table 6.1: Computational expense of SIMP.

| | Compliance | Average | Max |
|-------|------------|---------|--------|
| Time | 27.4 s | 26.6 s | 34.1 s |
| Iter. | 100 | 100 | 100 |

It is interesting to note that the computational expense of solving the average temperature minimization problem was lower than that of the compliance minimization problem. Though all algorithms performed the same number of outer loop optimization iterations, they performed different number of inner loop iterations in the MMA subroutine. For this reason, a direct comparison of optimization expense cannot be established. While it is clear that the optimization problems took roughly the same amount of time to solve, this similarity may be a fact of the small problem size. Validating these observations use larger scale problems is left a topic of future work. To culminate this investigation, the best performing designs are presented in

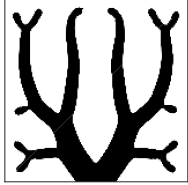
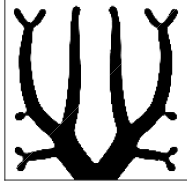
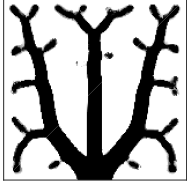
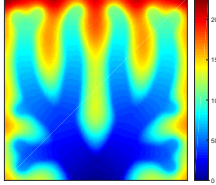
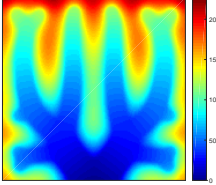
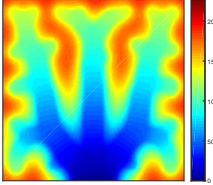
Fig. 6.2.

Table 6.2: Summary of SIMP results.

| | Compliance Solution | sum(T) Solution | $\ T\ _p$ Solution |
|------------|---|--|---|
| |  |  |  |
| |  |  |  |
| Compliance | 1997 | 2083 | 2250 |
| sum(T) | 2.6e6 | 2.7e6 | 2.9e6 |
| mean(T) | 101 C | 104 C | 112 C |
| $\ T\ _p$ | 453 C | 462 C | 416 C |
| max(T) | 226 C | 231 C | 191 C |

In this table, the best designs obtained using the SIMP algorithm only are presented. Each of the final topologies are evaluated across all objective functions to highlight similarities and differences. It is clear from this data that the SIMP algorithm does not obtain the best solution for a given objective when optimizing the given objective. The best optimized objectives are shown in bold. Take for example the temperature sum solution. A lower temperature sum was found when using the compliance objective function. However, when using the p-norm approximation, significantly reduced temperatures on the domain were found. This inconsistency in the results are corrected when using a hybrid method to pick a good starting topology for the SIMP method, demonstrating the multi-modal nature of these topology optimization problems.

Table 6.3: Summary of Hybrid results.

| | Compliance Solution | sum(T) Solution | $\ T\ _p$ Solution |
|------------|---|--|---|
| |  |  |  |
| |  |  |  |
| Compliance | 1920 | 1933 | 2180 |
| sum(T) | 2.5e6 | 2.5e6 | 2.8e6 |
| mean(T) | 96 C | 96 C | 109 C |
| $\ T\ _p$ | 424 C | 426 C | 410 C |
| max(T) | 227 C | 221 C | 189 C |

Once again the best optimized objective values are shown in bold. The temperature sum solution now obtains the same value as the compliance solution. This is due to the similarity between the two objectives, the compliance objective is a non-linear scaling to the temperature sum. This non-linearity affects the solution, which has slightly different scores. The p-norm solution was improved with the hybrid strategy and still dominates its objective. This is promising for use in optimization, but this simple objective may not accurately reflect the objective of the system. A strategy to address this is presented in the following chapter. It appears that obtaining consistent results with topology optimization is difficult. This issue can be helped using better starting points for the SIMP algorithm, however, the non-convex nature of this design problem causes convergence to local optima.

CHAPTER 7

TEMPERATURE CONSTRAINTS

In the previous chapters, generative algorithms were used to augment the SIMP algorithm for the heat spreader design problem. In the last chapter, alternative formulations for topology optimization were investigated to demonstrate the importance of using true objectives for optimization. Though temperature minimization is often the goal of a heat spreader design problem, it may not be the true goal of the overall system that includes the heat spreader. In power electronics applications, signal integrity is often as important as system loss characteristics. Having the minimum temperature on all of the domain may not present a system-level optimum, where directed heating may help increase signal integrity. Yet, there still exists a temperature a maximum temperature in which electrical components operate. Here, I propose using temperature constraints to satisfy heat spreader requirements, and using an objective that aligns better with the overall system performance. To explore the use of temperature constraints, the same design problem will be used once again.

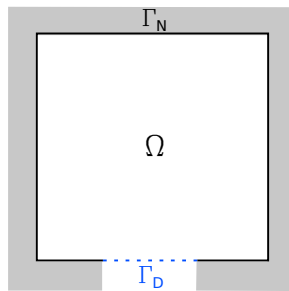


Figure 7.1: Homogeneously heated design domain.

In this study, the problem formulation is changed to include temperature as a constraint instead of as an objective. This allows us to use a higher-level system performance objective, while still ensuring the domain does not exceed a maximum allowed temperature. Consider the following formulation:

$$\begin{aligned}
& \underset{\mathbf{x}}{\text{minimize}} && \Theta \\
& \text{subject to} && V(\mathbf{x}) \leq V_p \\
& && R(\mathbf{x}) \geq R_p \\
& && T(\mathbf{x}) \leq T_p
\end{aligned} \tag{7.1}$$

The temperature constraint above requires that the temperature of an element, $T(\mathbf{x})$, is no higher than a prescribed value, T_p . To enforce this temperature constraint there are several options; these will be discussed in detail in this chapter and numerical results will be presented.

7.1 Temperature Constraint Variations

To enforce the temperature constraint, three constraint formulations are investigated. The first of which is to directly enforce one constraint per element to ensure no element violates the temperature constraint. An alternative to this may be to select a few point on the domain that have a high likelihood of high temperatures and constrain those points. In addition to point constraints, a single global constraint may be proposed to enforce a maximum temperature on the domain. These three constraint formulations are presented in Table 7.1.

Table 7.1: Alternatives temperature constraint formulations.

| Distributed Constraint | Selective Constraint | Global Constraint |
|--|---|---------------------|
| $T_i(x) \leq T_p \quad \forall i = [1, n_{\text{el}}]$ | $T_i(x) \leq T_p \quad \forall i \in \mathcal{S}$ | $\ T(x)\ \leq T_p$ |

The distributed constraint is enforced for all n_{el} elements on the domain. The selective constraint is enforced on all elements in some subset, \mathcal{S} . The global constraint may be approximated as a norm, similar to the examples in the previous chapter. Before presenting numerical results, I would like to discuss the implications of each temperature constraint in more detail.

The distributed constraint is the most simple to conceptualize. There exists one constraint per finite element, and enforcing this constraint will ensure that no element temperature exceeds the maximum temperature. Using the

MMA algorithm, enforcing such a large set of constraints is possible. However, implementing this constraint is computationally expensive. To obtain the sensitivity of the design variable to a single constraint, an adjoint problem must be solved. Solving for the adjoint vector for each constraint requires solving a separate linear system of equations for each constraint. This will decrease the speed of the topology optimization by the order of the number of elements per iteration. For this reason, the alternative formulations have been presented.

The next most simple method may be to enforce the constraint in locations where high temperature is likely. This strategy makes sense in power electronics applications where the temperature will be highest on the components, and enforcing this constraint on the elements of the components will prevent temperature failure with confidence. Physically speaking, temperature constraints only exist for the components on the board, so this constraint makes sense. This rationale may not be generalizable for all problems. This strategy sounds promising, however, there may still be a large number of constraints to enforce depending on the number of elements which represent these components.

The third constraint is a p-norm approximation of the maximum temperature on the domain. This strategy is appealing because it only requires the solution to one adjoint equation to obtain the sensitivities of the constraint. Choosing a large p-value for the norm approximates the max function, but makes the derivative ill-behaved. Choosing a small p-value results in an inaccurate description of the maximum temperature. Successfully using this constraint requires careful consideration of its implementation. A wiser approach may be to enforce one p-norm constraint per heating device, such as in power electronics application.

7.2 Numerical Case Study

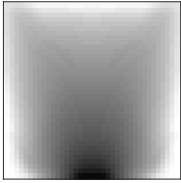
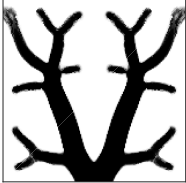
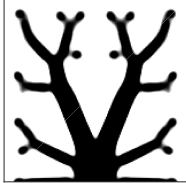
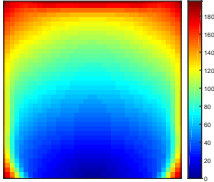
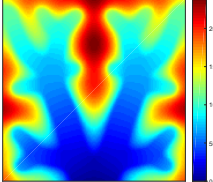
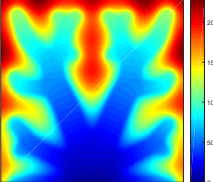
To show the effects of the different constraint strategies, they will each be enforced in a compliance minimization problem. Compliance is used here to decrease the computational burden of the design problem. Since the smallest maximum temperature obtained in the last chapter was 191°C , that will be enforced as the temperature constraint here. The idea is to use the temper-

ature constraint to force the MMA algorithm to find a topology near our current best design. The selective temperature constraint will be enforced at several key locations where temperature where often the highest. For the purpose of this study, the top left and right corners are constrained. The global constraint uses a p-value of 50. To increase the accuracy of the p-norm approximation, the constraint is enforced in the following manner:

$$\left\| \left\| \frac{T_{max}}{T_p} - 1 \right\| \right\|_p. \quad (7.2)$$

This normalization of the maximum temperature prevents raising large number to large powers. This allows using a larger p-norm. In addition, this prevents additional error which may be incurred from summing all of the temperatures on the domain. Recall that there exists a temperature at all finite element nodes, aggregating hundreds of thousands of temperatures introduces more error. The solutions to these problems are presented in Table 7.2.

Table 7.2: Summary of results for different constraints

| | Distributed Constraint | Selective Constraint | Global Constraint |
|------------|---|---|---|
| |  |  |  |
| |  |  |  |
| Compliance | 1859 | 2161 | 2057 |
| $\ T\ _p$ | - | - | 421 |
| $\max(T)$ | 208 | 239 | 228 |

There a few items to note from these results. The domain resolution was reduced for the distributed constraint problem. Solving this problem presents an unreasonable computational expense, and the simplified results are pre-

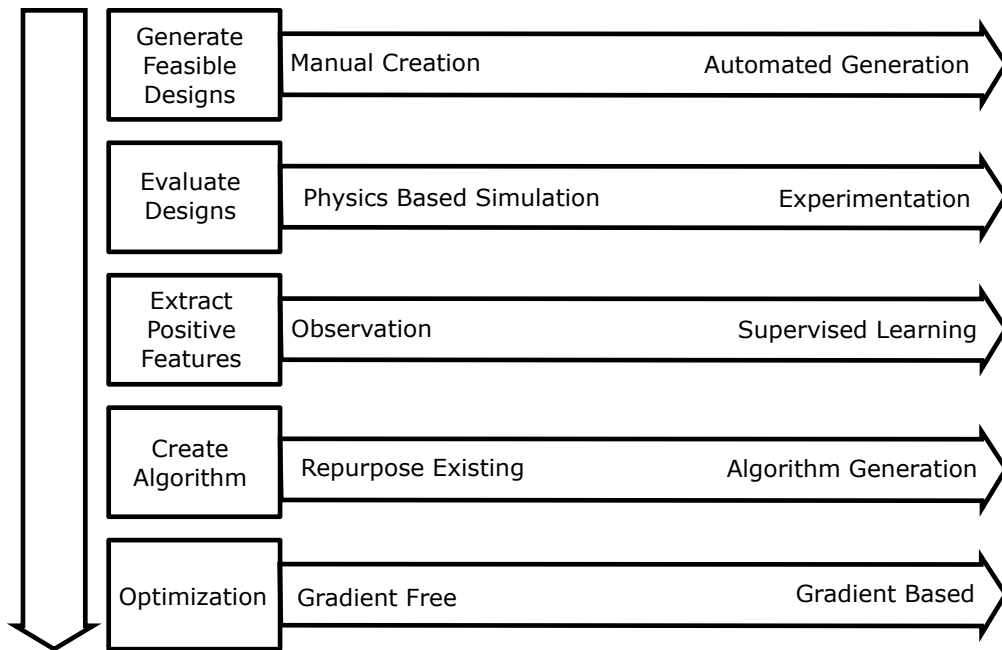
sented here for completeness. The selective constraint solution reduces the temperature successfully at the constraint points, however, temperature increases are observed at other parts of the domain. This same tendency was observed when varying the number and location of temperature constraints. The global constraint did not succeed at reducing the domain temperature to satisfy the constraint. Furthermore, the approximated maximum temperature was twice the actual maximum temperature. This is similar to the result seen when using the p-norm as an objective. As a constraint, it becomes important to obtain a true temperature. Normalizing this constraint as discussed did not reduce sufficiently the error of the norm approximation. Investigating an accurate measure for the maximum temperature on the domain is a critical factor in using topology optimization in multi-physics design with heat transfer, and is an important topic for ongoing work. Generative algorithms may alleviate some of these numerical issues when coupled to a gradient-free optimization algorithm. However, there will be an increased computational expense to find a feasible design.

CHAPTER 8

CONCLUSION

8.1 Thesis Summary

This thesis focused on improving topological design using generative algorithms. A framework was presented to create structure in the infinite space of design.



This framework includes an investigation of optimal heat spreader designs in literature. Optimal designs were observed to be dendritic in structure. The verification of this structure using SIMP topology optimization was conducted and used in comparison. The dendritic nature of optimal heat spreaders was used as a target for generative algorithm replication. The space colonization algorithm was chosen to create dendritic patterns for use as an abstract design representation for topology optimization.

One of the weaknesses of homogenization topology optimization methods is the lack of global convergence properties for the algorithms. To address this, a generative design method was proposed coupling the global search properties of the genetic algorithm with the local convergence properties of the MMA algorithm. Namely, a generative algorithm rule-set was optimized using a genetic algorithm to produce dendritic structures. These structures were evaluated and optimized by the GA. The final solution of the optimization is used as a starting point for the SIMP algorithm, where local search strengths can be utilized to fine tune the designs. The results obtained using this strategy frequently outperformed the typical homogeneous implementation of the SIMP method. Furthermore, it did so for various problem formulations that relate to cooling requirements. Another weakness that branches from local convergence is the inability of the optimization algorithms to obtain the best solution for a given objective. For example, the compliance minimization solution had a lower average temperature than the average temperature minimization solution. This was remedied somewhat using the hybrid optimization approach.

Further investigating the parameterization for design, it was noted that heat spreaders are used to reduce temperatures on a domain to a threshold. This motivated an exploration of temperature-constrained topology optimization problems. This a difficult problem to solve given the strict temperature requirements that must be enforced at all locations on the domain. As mentioned in the last chapter, generative algorithms may offer an alternative when used with gradient-free methods. Using a gradient-free method eliminates the need to calculate difficult gradients, such as those between lumped and continuum parameters. Further investigation of this problem formulation is left as a topic of future study.

8.2 Future Work

The results presented in this thesis resulted from a slow manual investigation, with slow manual tuning. With the investment of time, a consistent improvement in design performance was observed. In practice, engineers may not have sufficient time to tune their models and algorithms to have confidence in a result. This motivates a further investigation into an automated

generative design methodology that is driven by design data. The following list contains several key areas of future work to realize the full potential of a generative design methodology.

- Applying the generative design methodology to other forms of structural designs (e.g. mechanics, magnetics, fluid flow).
- Investigating the use of the generative design methodology to systems that can be represented by heterogeneous graphs (e.g. circuit or powertrain architecture with distinct component types).
- The automation of the generative design methodology in pattern recognition and algorithm creation for abstract design representation in optimization.
- Investigating the usefulness of problem formulations in capturing system requirements.

It is my hope that this thesis presents how simple topology optimization methodologies can be to implement. Furthermore, it is my hope that this demonstrates how simple it is to use a generative algorithm to find a good starting topology for conventional topology optimization methods. Using both gradient-free and gradient-based methods shows promise in addressing some of the known issues with homogenization-based approaches.

REFERENCES

- [1] A. Khetan, D. J. Lohan, and J. T. Allison, “Managing Variable-Dimension Structural Optimization Problems Using Generative Algorithms,” *Structural and Multidisciplinary Optimization*, vol. 50, no. 4, pp. 695–715, 2015.
- [2] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [3] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*, 2006.
- [4] A. Lindenmayer, “Development Algorithm for Multicellular Organisms: A Survey of L-Systems,” *Journal of Theoretical Biology*, vol. 54, pp. 3–22, 1975.
- [5] W. Stephen, “Statistical Mechanics of Cellular Automata,” *Rev. Mod. Phys.*, vol. 55, no. 3, pp. 601 – 644, 1983.
- [6] M. Gardner, “The Fantastic Combination of John Conway’s New Solitaire Game of ”Life” ,” *Scientific American*, vol. 223, pp. 120–123, 1970.
- [7] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz, “Modeling and Visualization of Leaf Venation Patterns,” in *ACM Transactions on Graphics*, vol. 24, no. 3, 2005, pp. 702–711.
- [8] T. Sachs, “The Control of Patterned Differentiation of Vascular Tissues,” *Advances in Botanical Research*, vol. 6, pp. 151–262, 1981.
- [9] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design*. Cambridge University Press, 2000.
- [10] A. Messac, *Optimization in Practice with Matlab*. Cambridge University Press, 2015.

- [11] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, “Unshackling Evolution: Soft Robotics with Multiple Materials and a Powerful Generative Encoding,” in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation Conference*, ser. GECCO '13. ACM, July 2013, pp. 167–179.
- [12] R. Fletcher and M. Powell, “A Rapid Convergent Descent Method for Minimization,” *The Computer Journal*, vol. 6, no. 2, pp. 163–168, 1963.
- [13] C. Audet and J. Dennis Jr, “Analysis of Generalized Pattern Searches,” *SIAM Journal on Optimization*, vol. 13, no. 3, pp. 889–903, 2003.
- [14] A. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2003.
- [15] P. Hajela, E. Lee, and C. Lin, “Genetic Algorithms in Structural Topology Optimization,” in *Topology design of structures*. Springer, 1993, pp. 117–133.
- [16] K. Deb and S. Gulati, “Design of Truss-Structures for Minimum Weight Using Genetic Algorithms,” *Finite Elements in Analysis and Design*, vol. 37, no. 5, pp. 447–465, May 2001.
- [17] T. Hagishita and M. Ohsaki, “Topology Optimization of Trusses by Growing Ground Structure Method,” *Structural and Multidisciplinary Optimization*, vol. 37, no. 4, pp. 377–393, January 2009.
- [18] V. Venkayya, “Design of Optimum Structures,” *Computers & Structures*, vol. 1, no. 1, pp. 265–309, Aug 1971.
- [19] L. A. Schmit and B. Farshi, “Some Approximation Concepts for Structural Synthesis,” *AIAA Journal*, vol. 12, no. 5, pp. 692–699, 1974.
- [20] M. Dobbs and R. Nelson, “Application of Optimality Criteria to Automated Structural Design,” *AIAA Journal*, vol. 14, no. 10, pp. 1436–1443, 1976.
- [21] D. Goldberg and M. Samtni, “Engineering Optimization via the Genetic Algorithms,” *Computers and Structures*, vol. 40, pp. 1321–1327, 1991.
- [22] S. Rajeev and C. Krishnamoorthy, “Discrete Optimization of Structures Using Genetic Algorithms,” *Journal of Structural Engineering*, vol. 118, no. 5, pp. 1233–1250, May 1992.
- [23] H. Rahami, A. Kaveh, and Y. Gholipour, “Sizing, Geometry and Topology Optimization of Trusses via Force Method and Genetic Algorithm,” *Engineering Structures*, vol. 30, no. 9, pp. 2360–2369, September 2008.

- [24] M. Giger and P. Ermanni, “Evolutionary Truss Topology Optimization Using a Graph-Based Parameterization Concept,” *Structural and Multidisciplinary Optimization*, vol. 32, no. 4, pp. 313–326, 2006.
- [25] S. Rajan, “Sizing, Shape, and Topology Design Optimization of Trusses Using Genetic Algorithm,” *Journal of Structural Engineering*, vol. 121, no. 10, pp. 1480–1487, October 1995.
- [26] R. J. Balling, R. R. Briggs, and K. Gillman, “Multiple Optimum Size/Shape/Topology Designs for Skeletal Structures Using a Genetic Algorithm,” *Journal of Structural Engineering*, vol. 132, no. 7, pp. 1158–1165, July 2006.
- [27] A. Kaveh and K. Laknejadi, “A Hybrid Evolutionary Graph-Based Multi-Objective Algorithm for Layout Optimization of Truss Structures,” *Acta Mechanica*, vol. 224, no. 2, pp. 343–364, November 2013.
- [28] C. M. Chilan, D. R. Herber, Y. K. Nakka, S.-J. Chung, J. T. Allison, J. B. Aldrich, and O. S. Alvarez-Salazar, “Co-Design of Strain-Actuated Solar Arrays for Precision Pointing and Jitter Reduction,” in *In AIAA 2016 Science and Technology Forum and Exposition*, Jan 2016.
- [29] M. P. Bendsoe and N. Kikuchi, “Generating Optimal Topologies for Structural Design Using a Homogenization Method,” *Computational Methods in Applied Mechanical Engineering*, vol. 71, no. 2, pp. 197–224, 1988.
- [30] X. Huang, S. Zhou, Y. Xie, and Q. Li, “Topology Optimization of Microstructures of Cellular Materials and Composite for Macrostructures,” *Computational Materials Science*, vol. 67, pp. 197–407, 2013.
- [31] M. P. Bendsoe, “Optimal Shape Design as a Material Distribution Problem,” *Structural Optimization*, vol. 1, no. 4, pp. 193–202, 1989.
- [32] M. Stolpe and K. Svanberg, “An Alternative Interpolation Scheme for Minimum Compliance Optimization,” *Structural and Multidisciplinary Optimization*, vol. 22, no. 2, pp. 116–139, 2001a.
- [33] O. Sigmund, “On the Design of Compliant Mechanisms Using Topology Optimization,” *Mechanics of Structures and Machines*, vol. 25, no. 4, pp. 493–524, 1997.
- [34] T. Bruns and D. Tortorelli, “Topology Optimization of Non-Linear Elastic Structures and Compliant Mechanisms,” *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 26–27, pp. 3443–3459, 2001.

- [35] J. K. Guest, J. Prevost, and T. Belytschko, “Achieving Minimum Length Scale in Topology Optimization Using Nodal Design Variables and Projection Functions,” *International Journal of Numerical Methods in Engineering*, vol. 61, no. 2, pp. 238–254, 2004.
- [36] K. Svanberg and H. Svard, “Density Filters for Topology Optimization Based on the Geometric Harmonic Means,” in *10th World Congress on Structural and Multidisciplinary Optimization*, Orlando, U.S.A., May 2013.
- [37] O. Sigmund and K. Maute, “Topology Optimization Approaches: A Comparative Review,” *Structural and Multidisciplinary Optimization*, vol. 48, no. 6, pp. 1031–1055, 2013.
- [38] A. Gersborg-Hansen, M. P. Bendsoe, and O. Sigmund, “Topology Optimization of Heat Conduction Problems Using the Finite Volume Method,” *Structural and Multidisciplinary Optimization*, vol. 31, pp. 251–259, Mar 2006.
- [39] N. de Kruijf, S. Zhou, Q. Li, and Y.-W. Mai, “Topological Design of Structures and Composite Materials with Multiobjectives,” *International Journal of Solids and Structures*, vol. 44, pp. 7092–7109, 2007.
- [40] C. Zhuang, Z. Xiong, and H. Ding, “A Level Set Method for Topology Optimization of Heat Conduction Problems Under Multiple Loading Cases,” *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 1074–1084, Jan 2007.
- [41] E. M. Dede, “Optimization and Design of a Multipass Branching Microchannel Heat Sink for Electronics Cooling,” *ASME Journal of Electronic Packaging*, vol. 134, no. 4, p. 041001, Dec 2012.
- [42] Y. Chen, S. Zhou, and Q. Li, “Multiobjective Topology Optimization for Finite Periodic Structures,” *Computers & Structures*, vol. 88, pp. 806–811, 2010.
- [43] F. H. Burger, J. Dirker, and J. P. Meyer, “Three-Dimensional Conductive Heat Transfer Topology Optimization in a Cubic Domain for the Volume-To-Surface Problem,” *International Journal of Heat and Mass Transfer*, vol. 67, pp. 214–224, Dec 2013.
- [44] T. E. Bruns, “Topology Optimization of Convection-Dominated, Steady-State Heat Transfer Problems,” *International Journal of Heat and Mass Transfer*, vol. 50, no. 15-16, pp. 2859–2873, 2007.
- [45] A. Iga, S. Nishiwaki, K. Izui, and M. Yoshimura, “Topology optimization for thermal conductors considering design-dependent effects, including heat conduction and convection,” *International Journal of Heat and Mass Transfer*, vol. 52, no. 11-12, pp. 2721–2732, 2009.

- [46] G. H. Yoon, “Topological Design of Heat Dissipating Structure with Forced Convective Heat Transfer,” *Journal of Mechanical Science and Technology*, vol. 24, no. 6, pp. 1225–1233, 2010.
- [47] E. M. Dede, P. Schmalenberg, T. Nomura, and M. Ishigaki, “Design of Anisotropic Thermal Conductivity in Multilayer Printed Circuit Boards,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 5, no. 12, pp. 1763–1774, Dec 2015.
- [48] A. Bejan, I. Dincer, S. Lorente, A. F. Miguel, and A. H. Reis, *Porous and Complex Flow Structures in Modern Technology*. Springer Science, 2004.
- [49] S. Salakij, J. A. Liburdy, D. V. Pence, and M. Apreotesi, “Modeling in Situ Vapor Extraction During Convective Boiling in Fractal-Like Branching Microchannel Networks,” *International Journal of Heat and Mass Transfer*, vol. 60, no. 0, pp. 700–712, May 2013.
- [50] D. Heymann, D. Pence, and V. Narayanan, “Optimization of fractal-like branching microchannel heat sinks for single-phase flows,” *International Journal of Thermal Sciences*, vol. 49, no. 8, pp. 1383–1393, 2010.
- [51] M. Meinhardt, “Morphogenesis of Lines and Nets,” *Differentiation; Research in Biological Diversity*, vol. 6, no. 2, pp. 117–123, aug 1976.
- [52] Y. Rodkaew, S. Siripant, C. Lursinsap, and P. Chongstitvatana, “An Algorithm for Generating Vein Images for Realistic Modeling of a Leaf,” in *Computational Mathematics and Modeling*, 2002.
- [53] A. Runions, B. Lane, and P. Prusinkiewicz, “Modeling Trees With a Space Colonization Algorithm,” in *Eurographics Workshop on Natural Phenomena*, 2007, pp. 63–70.