AN ABSTRACT OF THE THESIS OF

Connor Dokken for the degree of Master of Science in Mechanical Engineering presented on November 28, 2017.

Title: Potential of Genetic Algorithms for Design of Liquid Cooled Heat Sinks Fabricated Using Additive Manufacturing

Abstract approved: _____

Brian M. Fronk

The advancement of new manufacturing techniques such as additive manufacturing enable significantly greater freedom of design compared to conventional techniques. Of significant interest is what improvements could be made in the design of cold plates and heat sinks for electronics cooling. Greater design freedom could potentially enable new designs that reduce thermal resistance and hydraulic resistance, enabling the usage of higher power systems while maintaining an equivalent operation envelope. The potential of using genetic algorithms to produce optimum shapes for cold plate heat sinks is investigated in this thesis. A 50 × 10 mm area is established into grid space, with initial designs initiated by probability based on representative chip power distribution profiles. Two different power maps are used to develop optimized heat sink geometry, one being a generated symmetric map, and the other being a non-

symmetric map based on a real processor. After the initial designs are developed and performance results determined from computational fluid dynamics, an optimization algorithm from the literature is utilized. Solution ranking is based on the system entropy generation rate, which is dependent on solution pressure drop and thermal resistance. Crossover, mutation and elitism operations are used to create new generations of designs, eventually leading to an optimized solution. Optimized results are compared with a baseline straight finned cold plate to determine the advantages of designing for an unrestricted manufacturing process. Results indicate that the optimization methods reduced entropy generation rate by 26.4% and 21.7% for the symmetric and non-symmetric power maps respectively.

Potential of Genetic Algorithms for Design of Liquid Cooled Heat Sinks Fabricated Using Additive Manufacturing

by

Connor Dokken

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Presented November 28, 2017

Commencement June 2018

Master of Science thesis of Connor Dokken presented on November 28, 2017

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Connor Dokken, Author

ACKNOWLEDGEMENTS

I would like to first thank Dr. Brian Fronk. He has been an excellent advisor and guide throughout my graduate studies. Not only has he guided me to be a better researcher, student and engineer, he has actively encouraged my career interests and goals. His help allowed me to become involved with the fellowship opportunities that would guide my thesis research, and establish my future career. Furthermore, he was readily available and active whenever I needed assistance. Thank you so much Dr. Fronk.

Next, I would like thank each member of my research lab: Paul Armatis, Matthew Hyder, Tabeel Jacob, Saad Jajja, Michael Vanderputten and Kyle Zada. Their persistence and help was instrumental to my success throughout classes and in my research. They always made time, and helped guide me as I was running through all of my ideas in my research. I have made so many fond memories in our time together, and I will miss each of and every one of you as we move on from school.

My family and friends are another group in need of thanks. Your continued support in my research and assistance in continued relocation was invaluable. You have helped me through any hard times, and I couldn't have gotten this far without any of you.

Finally, I would like to extend my thanks to Intel Corporation. Their fellowship offer has made my time in graduate school possible, as well as providing many work opportunities. Additionally, the continued support of my co-workers and other persons at the company has allowed me to finish the task of completing this research. I look forward to a long and successful career with the company.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued) Page

TABLE OF CONTENTS (Continued) Page

LIST OF FIGURES

Page

LIST OF FIGURES (Continued) Page

LIST OF FIGURES (Continued) Page

LIST OF FIGURES (Continued) Page

LIST OF FIGURES (Continued) Page

LIST OF TABLES

Page

Nomenclature

A	Area (m^2)
E	Energy (J)
F	Force (N)
h	Specific Enthalpy (J/kg)
k	Thermal Conductivity (W/m-K)
$\dot{m}$	Mass flow rate (kg/s)
P	Pressure (Pa)
$\dot{Q}$	Power (W)
R	Thermal resistance (K/W)
s	Specific Entropy (J/kg-K)
S_{gen}	Entropy Generation Rate (W/K)
T	Temperature (K)
$\bar{v}$	Transported variable (viscosity-related) (kg/s-m)
$\dot{W}$	Work rate (W)

Greek

α	Thermal diffusivity (m^2/s)
μ	Dynamic viscosity (Pa-s)
ρ	Density (kg/m^3)

Fluent Model Constants

C_{b2}

G_v

S_h

σ_v

$S_{\bar{v}}$

Y_v

Subscripts

b	Base
D	Drag
eff	Effective
∞	Ambient
q	Mass-averaged variable in energy model

Acronyms

AM – Additive Manufacturing

CFD – Computational Fluid Dynamics

CSV – Comma-Separated Values file type

DM – Decision Maker

EBM – Electron Beam Melting

GA – Genetic Algorithm

GCI – Grid Convergence Index

IC – Integrated Circuit

LENS – Laser Engineered Net Shaping

MECS – Microtechnology-based Energy and Chemical Systems

MOCO – Multi-Objective Combinatorial Optimization

PBIL – Population-Based Incremental Learning

PSO – Particle Swarm Optimization

SLM – Selective Laser Melting

SLS – Selective Laser Sintering

SQP – Sequential Quadratic Programming

STL – Standard Tessellation Language file type

1. Introduction

1.1 Background

Currently, the electronics industry is encountering growing challenges with thermal management. Heat densities are rapidly increasing, while at the same time the volume available for thermal management solutions are decreasing. The advancement of manufacturing techniques opens the opportunity to investigate different heat sink designs and design methodologies. The primary opportunity is that more of the 2D and 3D design space is available for modification with the use of additive manufacturing (AM), compared to traditional manufacturing technologies such as extrusion, stamping or forming. Thus, the objective of this thesis is to explore the use of genetic algorithms for developing different geometries of liquid cooled heat sinks that could only be fabricated using additive manufacturing techniques and perform better than conventional technologies.

1.2 Previous Investigations

A few studies have begun to explore the potential for additive manufacturing in electronics cooling applications. Notably, Bornoff and Parry (2015), have explored optimization by "growing" a heat sink based on an initial plate and uniform heat source in the center of the area. The problem is initialized by running a simulation, and calculating the temperature profile across the base. The hottest part of the base material is determined, and additional material is added at that point, creating the basis of a new fin. Another simulation is run, the new hot point is determined and more material is added. This process is repeated until the design space is filled (with the material addition done symmetrically across the center). Additional growth periods from the base are performed until convergence is reached. The final device is shown in Figure

1.1. They found that the “grown” heat sink showed about a 20% reduction in thermal resistance compared with a standard extruded heat sink. It is worth noting that the standard heat sink has much larger fins than the minimum feature size of the “grown” device. To provide a better comparison, they also built an unconstrained finned heat sink to compare. This unconstrained design has no fixed aspect ratio and a minimal feature size the same as the “grown” heat sink. With this comparison, they found that the unconstrained design actually provided 5% lower thermal resistance than the “grown” heat sink.

Figure 1.1: Bornoff and Parry’s (2015) organically grown heat sink

They then continued this research by studying the effects of removing material from a traditional finned heat sink, to see if they can reduce system mass with a minimal trade-off in thermal resistance. Utilizing the unconstrained optimized heat sink used in the previous study (Bornoff & Parry, 2015), Bornoff *et al.* (2016) analyzed the heat sink, and began removal of fin material from low temperature zones. They did this in both an unrestricted manner, and also by limiting removal to the top surfaces of the device. The tradeoff in mass reduction to the thermal

resistance increase was measured and plotted, which always resulted in an exponential curve. In general, the unrestricted subtraction was consistently more effective, with a slower growing exponential curve then the top surface restriction. Interestingly, when they allowed unrestricted subtraction, framing in the fins began to occur. This is when rectangular shapes are removed from the center area of the fin, and is visible in Figure 1.2.

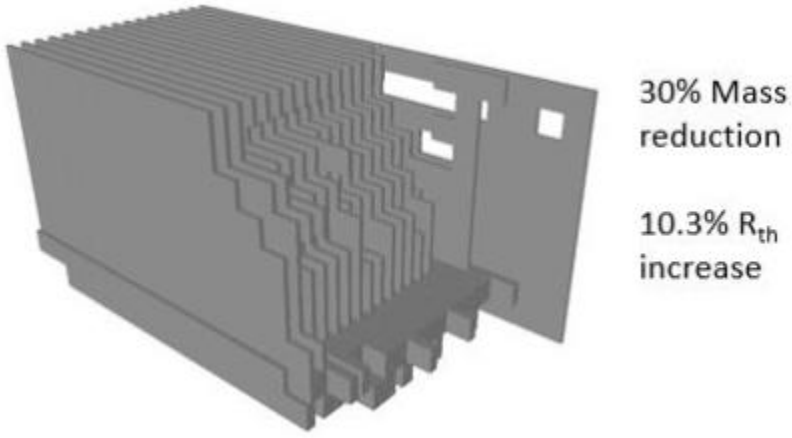

Figure 1.2: Bornoff and Parry's (2016) subtractive design heat sink, showing mass tradeoffs can be made without high impact on thermal resistance

Wu *et al*. (2016) performed research with genetic algorithms to provide optimization for a heat pipe design in a heat sink. They performed a thermal analysis of a commercially available heat sink, which consisted of copper heat pipes carrying coolant (water), embedded in an aluminum block. After analysis of the baseline commercial heat sink was completed, an optimization algorithm technique was applied to improve the design. An initial area was established with an inlet and outlet, as well as a heat source, and the algorithm was allowed to randomly generate paths between the inlet and outlet. Once completed, the designs are evaluated and ranked, and then utilizing genetic algorithms, new generations were created and the process repeated. After a certain amount of iterations were completed, a second stage occurred that performed perturbations on the existing designs, then repeated the genetic algorithm process. The

findings showed that the first part of the optimization process resulted in a design that reduced peak temperature from 53.02 °C to 46.10 °C, in exchange for an increase of inlet force from 0.56 N to 1.63 N. Similarly, the second part of the optimization process got a resulting peak temperature of 44.85 °C with an inlet force of 1.3 N. The results show the benefits of optimization processes, but also shows that the results achieved are not always free, in this case with increased internal pressure drop. Their solution is provided in Figure 1.3.

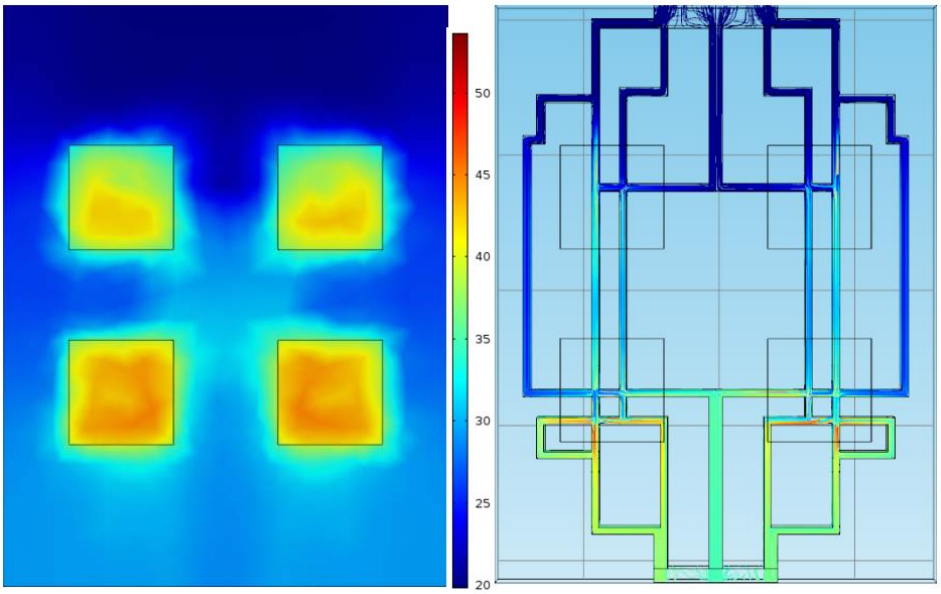

Figure 1.3: Wu *et al.*'s (2016) genetic algorithm heat sink

While Bornoff and Parry's (2016) initial study was interesting, it did show that the growth techniques used did not provide a significant benefit in comparison to the unconstrained optimized straight finned heat sink. Yet, Wu *et al*. found that utilizing genetic algorithm techniques on a water heat pipe heat sink yielded an 15.4% reduction in thermal resistance in exchange for an 132% increase of force, which may be an acceptable trade-off for some high flux applications. This thesis aims to determine if other optimization methodologies will allow

an additive manufactured heat sink to provide greater performance than the traditional finned heat sink.

1.3 Utilized Technologies and Techniques

One of the objectives of this study is to determine the potential benefits of combining new manufacturing technologies with genetic optimization techniques. Additive manufacturing technology and its rapid growth serves as a primary motivator, opening up a greater design space than is available with traditional manufacturing and also enabling new methods to develop product designs.

1.3.1. Additive Manufacturing

There are a variety of different additive manufacturing techniques available; most commonly using 3D printing. 3D printing techniques for metals are predominantly powder based technologies, including Selective Laser Sintering (SLS), Electron Beam Melting (EBM) and Prometal (Wong & Hernandez, 2012). Many of these technologies build parts similarly to plastic printing techniques, as presented in Figure 1.4. These technologies are all effective, but each can have some minor effect on the final material characteristics. In general, the final printed material is highly similar to the original material stock, but differences are observed due to final grain arrangement. Densities reach up to 99.8% of the parent material, but mechanical properties can vary significantly depending on build method. In many cases, the printed material meets or exceeds the cast or wrought part, generally trading strength for ductility. (Murr *et al*., 2012) Other options utilizing technologies such as inkjet are coming to market with the ability to print metal, but none are yet commercially available ("XJet 3D," 2017). Since the majority of metal printing techniques rely on metal powders, their deployment is limited to specialized manufacturing facilities, due to the toxicity and hazards of the raw material powder.

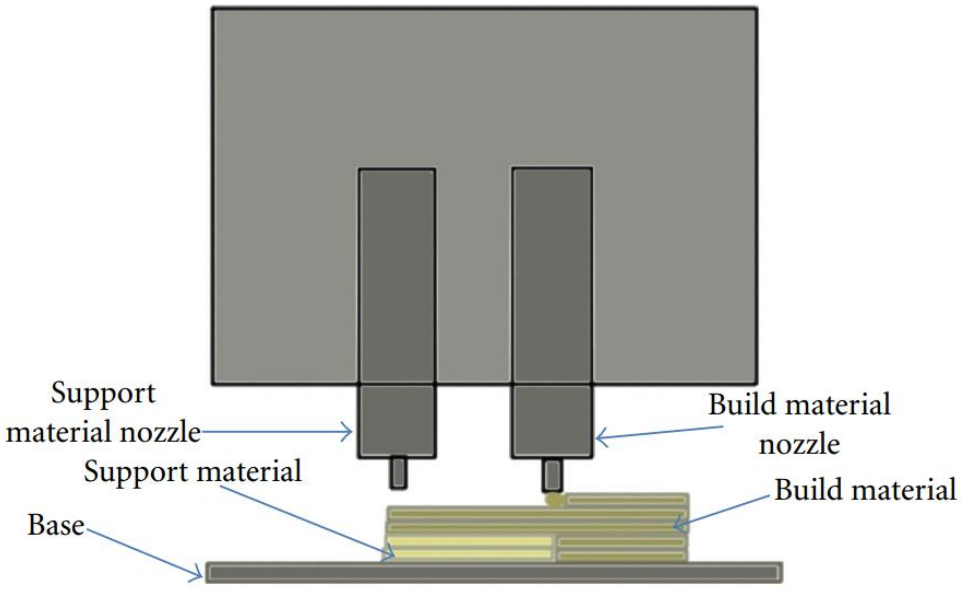

Figure 1.4: Diagram of a Fused Deposition Modeling plastic printing process (Wong & Hernandez, 2012)

The usage of additive manufacturing has a number of economic advantages compared to traditional manufacturing techniques such as casting or injection molding. AM allows the localization of manufacturing closer to production, reduction of transportation cost, simplified material storage, and elimination of custom fixturing. In general AM does not significantly benefit from economies of scale, thus, each final product has a fixed unit cost. In comparison, traditional manufacturing typically benefits greatly from economies of scale, to a point of diminishing returns. This means, that with smaller production sizes, AM may be the more cost effective manufacturing technique (Thomas & Gilbert, 2014). This is particularly intriguing as electronic device becoming increasingly customized, suggesting that heat sinks could be manufactured in low volumes for a tailored application for a particular customer.

After a certain number of units, the economies of scale dominate and traditional manufacturing techniques become the superior choice. The exact number of units in which traditional manufacturing overcomes AM is dependent on a number of factors. The technology

used, material, as well as size and complexity of the manufactured part among them. Thomas and Gilbert (2014) present results from Hopkinson and Dickens (2003) as well as Atzeni and Salmi (2012) to compare costs of AM to traditional techniques. Hopkinson and Dickens perform a study using a lever and cover, finding it took approximately 13,000 parts for injection molding costs to overtake AM. The results of this study are presented in Figure 1.5. Atzeni and Salmi investigated manufacturing of a landing wheel assembly for a 1:5 scale model plane. They found after 42 assemblies, the high-pressure die cast parts were cheaper to manufacture than the selective laser sintered part. These studies are useful in showing that estimates need to be performed based on the part made and technology currently available. (Thomas & Gilbert, 2014)

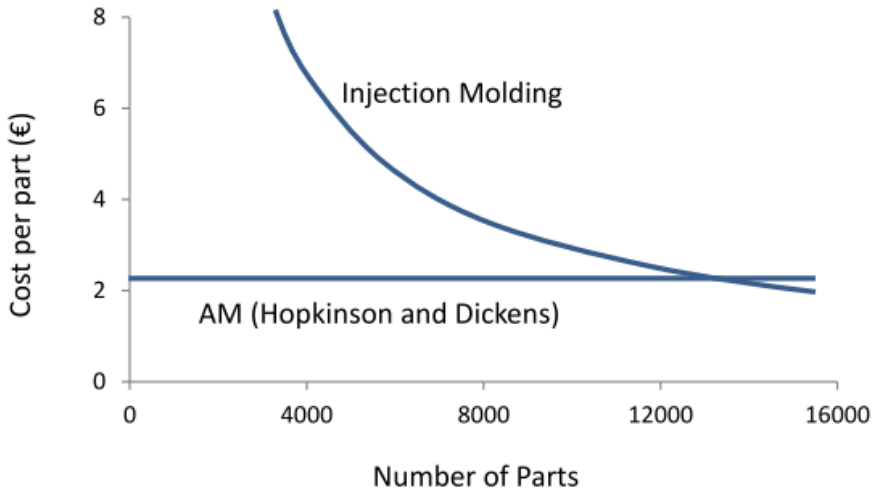

Figure 1.5: Example of the fixed costs of AM compared to the economies of scale of a traditional manufacturing technique. Analyzed by Hopkinson and Dickens (2003) and as reported by Thomas and Gilbert (2014)

1.3.2 Optimization Techniques

There are a number of different optimization techniques available for component design. Before deciding the best methodology, it is important to understand the building process in additive manufacturing. The machines lay material down layer by layer, and geometry can be defined by

a minimum feature size. These are often described by a cubic geometry, which is called a voxel in 3D space. If translated into 2D space, they create pixels, which are also referred to as bits. To represent a large area of 2D geometry, often a bit array can be used, which requires thought on the ideal methodology to use for optimization. One of the most promising techniques and the one used in this research is genetic algorithms (GA). Genetic algorithms are based on the idea of "survival of the fittest", in that the best solutions are allowed to be "parents" and produce a new generation of solutions (from a number of operators), which repeat the process until some convergence criterion. These algorithms are used to drive a solution based on known best solutions. In the case of the heat sinks, this may prove difficult, however, as there are no explicit results based on individual bit placement. Another option available is particle swarm optimization, which is similar to GAs, but based on the idea of "particle" position and velocity. The idea being that the "particle" has a given velocity based on changing values and the total velocity is influenced by the "swarm" of other particles, which result in dictating a new position each generation. This is unfortunately, difficult and awkward to implement for a discrete problem. (Eberhart & Shi, 1998)

When utilizing genetic algorithms, most problems are based on optimizing multiple opposing objectives. Typically, the results are plotted onto an X-Y axis, and form Pareto fronts. With multiple generations, the front should start moving closer to the axis until the system has reached convergence. The Pareto front methods are presented in Figure 1.6. Ranking the solutions on the front can be done in a few different ways, the most common involve creating a curve across the points. Any solution on the lowest curve are considered the best candidates, and will be used to create the new solution. There are alternative methods, including solution weighting, which uses the weights to rank each point, in order to determine what will be used to

generate the new generation. In a way, this actually turns the multi-objective optimization into a single-objective problem. (Gen & Cheng, 1999)

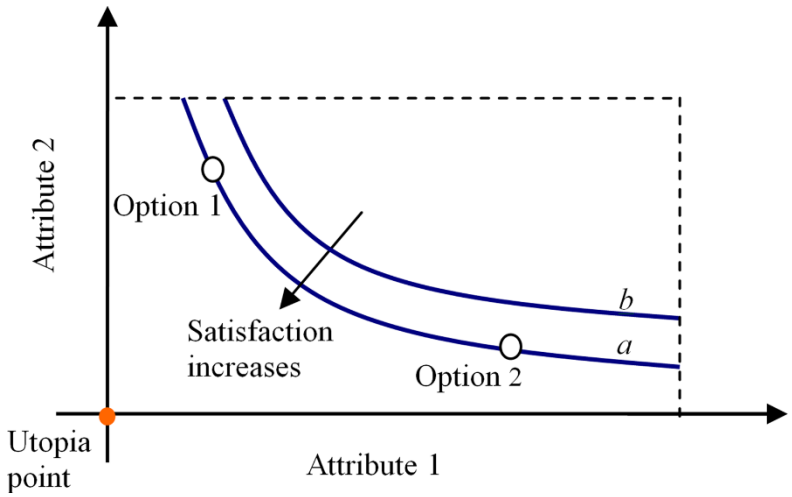

Figure 1.6: Pareto front optimization (Pandey, Mourelatos, & Nikolaidis, 2013)

Typically, the population size of each generation ranges from around 25 to 100 individuals, and can even be larger (Deb & Agrawal, 1999). This size is quite limiting for applications requiring intensive numerical analysis, as the computational time is significant. To get around this, some studies have used micro-genetic algorithms, which work with populations sizes of around 5 (Coello & Pulido, 2001). To allow an effective greater population variety, an external memory is used that is compared to solutions in the current generation. A few solutions in the current generation are selected (potentially based on ranking), and then a random solution in the memory is chosen. If the generational solution is superior to the one in the memory, that solution is kept in the generation and used to develop the next generation. Additionally, the generational solution replaces the solution in the memory, and is used in any future comparisons. On the other side, if the memory solution is the better solution, it replaces the generation solution

and is used to create the succeeding generation. The methodology used by Coello Coello and Pulido is presented in Figure 1.7.

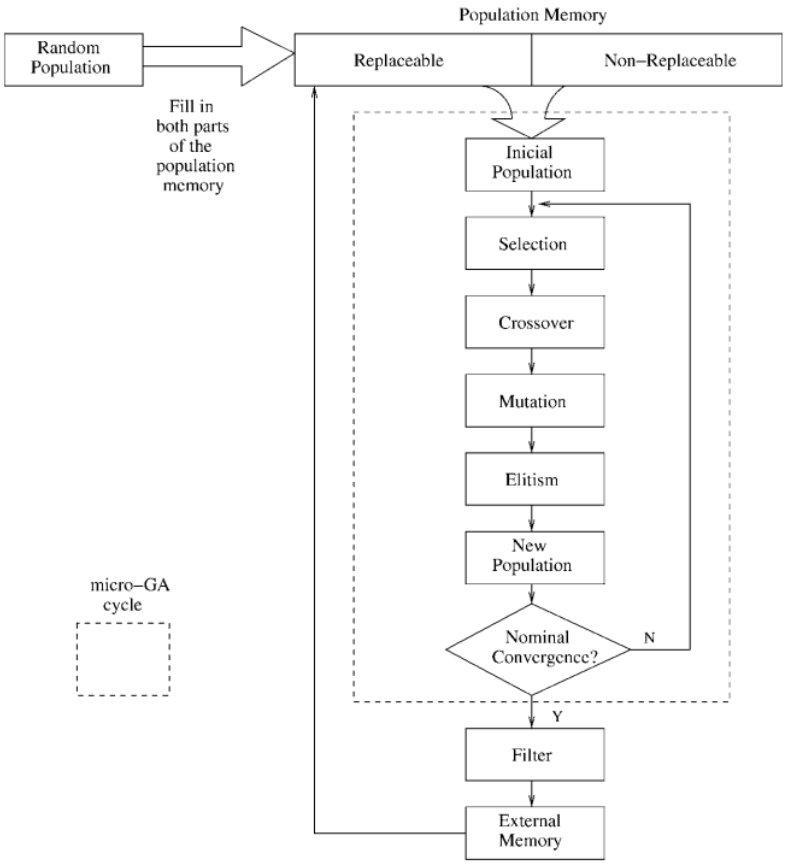

Figure 1.7: Micro-genetic algorithm overview of Coello Coello & Pulido (2001)

Finally, there are three typical operators that are used to create new generations; crossover, mutation and elitism. Crossover utilizes two parent solutions to create the child, by randomly taking information from the each of the parents and using it to set the child's bit data. Mutation operators takes a parent, and randomly changes information in it by some percentage to create a child. Often, mutation is combined with crossover operations to achieve the new children. Crossover and mutation are presented in Figure 1.8. Lastly, elitism brings the parent solution straight through to the new generation with no modification. (Chapman *et al.* , 1994)

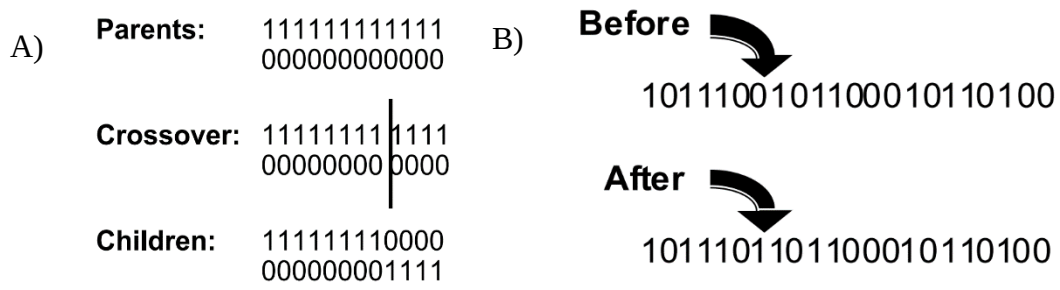

Figure 1.8: A) Crossover operation, B) Mutation operation (Chapman *et al*., 1994)

1.4. Electronics Cooling Design

There are a number of design options available for cooling electronics, depending on a variety of factors. These factors include criteria such as power density, space available, design thermal limits and cost. The most common available systems are (1) passive, (2) single-phase fluid active, (3) two-phase fluid active and (4) microchannel systems.

1.4.1. Heat Sink Design

Traditional heat sink design is typically based on uniform 2D profiles extended through a 3D space. These heat sinks are manufactured in few different ways, which are well described by Lee (1995): Stamping, where sheet metal is formed into the desired shape for the heat sink. While cost effective, it is limited in capability and should be limited to designs with low thermal density. Extrusion is fairly common for use in a number of systems, and is formed by machining a material stock to create fins. In some cases, the stock is crosscut to produce pin geometry. It is commonly used in both passive and active applications. Bonded heat sinks are extremely common in modern computing applications, and are created by manufacturing a base with slots, allowing fins to be placed in and retained by thermally conductive epoxy. Because of the greater fin to height ratio, as well as potential for more fins and greater surface area than extruded and stamped, they can be highly effective. Cast fins are usually utilized when developing pin fin geometry arrays, which can allow greater efficiency for systems such as those using

impingement flows. Finally, folded fins utilize sheet metal folding to create fin geometry which is then bonded to a base. These heat sinks allow efficiency in systems where bonded or extruded fins are not practical. While a few other designs are available, they are not as common as the manufacturing techniques and designs discussed. (Lee, 1995)

1.4.1.1 Passively Cooled Systems

Passively cooled systems are typically useful for systems with a low power density. The system relies on natural convection to manage device heat, either through designs such as an extruded finned base or through a heat spreader. Because of the limited heat transfer available through natural convection, it is not effective in dissipating higher powered system. Lee (1995) suggests these systems are effective from 5 to 50 watts, and suggests a batch of 10,000 will be priced at $0.50 each. While ineffective at thermal management, the heat sinks/spreaders are cost effective, allow reduction of electronic packaging height and eliminate the need for a noisy fan or pump (Lee, 1995). A bonded heat sink is shown in Figure 1.8.

Figure 1.9: Bonded finned heat sink (“Bonded Fin Image,” 2017)

1.4.1.2 Single-Phase Actively Cooled Systems

Single-phase actively cooled systems are among the most commonly used heat sink designs in electronic systems. The systems rely on pumps or fans to provide forced convection through the heat sink, which consists of a finned array or a fluid channel. Active systems work well for low to high heat flux designs. Tradeoffs are made with the passive system for the greater cooling, being both more costly and noisier than a passive system, an often more than acceptable exchange. For this system, it suggested that they can handle 10 to 160 watts, and cost from $10.00 to $20.00 each in a batch of 10,000.When compared to two-phase and microchannel systems, the standard active system benefits from significantly lower cost and engineering investment. (Lee, 1995)

1.4.1.3 Two-Phase Actively Cooled Systems

Two-phase systems are reliant on fluid to gas transitions to cool a system utilizing the large latent heat and high heat transfer coefficients available through boiling. However, these systems are highly complex, meaning that the cost can be significant. In addition to ensuring the system has enough heat flux to boil the fluid, it also needs to manage it to prevent burnout, as well as condensing the vapor to maintain the circulating flow. The benefit for managing this complexity is a high heat transfer rate capability (suggested to handle power from 100 to 150 W by Lee (1995), which is lower than some studies, which have tested to powers beyond 1000 W (Gillot *et al*., 2000). Two-phase systems have great theoretical performance for use in systems such as high-power servers. Before wider adoption, power density in commercial systems need to reach a point where the additional costs of the cooling devices necessitates the trade-off, estimated to be anywhere between $15.00 to $500 each. (Ghiaasiaan, 2008; Lee, 1995)

1.4.1.4 Microchannel Systems

Microchannel heat exchangers can utilize either single-phase or two-phase cooling methodologies. While arguably they can be included with the other systems categories, it is important to recognize the distinction. With microchannel systems, the scale of the device allows it to take advantage of scaling laws, achieving high heat transfer rates in a smaller area than traditional devices. A study by Kandlikar (2005) suggests a single phase system capable of 100,000 W/m^2-K. Assuming a 2 cm square device and a 20 K temperature difference, this approximates 800 W of power dissipation. This allows them to be very effective cooling devices, but this of course comes with cost. Microchannel devices are manufactured in a number of ways, some include laser cutting and welding, others use micromachining (Paul & Peterson, 1999). Lajevardi *et al.* (2011) suggest costs of up to $5000 in batches of 200 per year, reducing to $500 in batches of 10,000 per year. Additionally, if the designer wishes to implement two-phase cooling, the costs increase further.

1.5. Project Objectives

The overall objective of this research is to determine if heat sinks designed to take advantage of additive manufacturing techniques show advantages over heat sinks utilizing traditional manufacturing techniques. To compare effectiveness, the two objectives of interest are pressure drop and thermal resistance. To best rank the solutions, the entropy generation rate of the solution is utilized, which makes use of both the objectives, and essentially turns the system into a single-objective optimization problem. The solution system uses a volume of 52 mm × 50 mm × 12 mm of total volume, and 50 mm × 50 mm × 10 mm of fluid volume space. This size was specified based on a comparable cold plate analysis (Remsburg, 2007) Water is used as the working fluid as requested by industry discussion. The entropy generation value is then

compared to a traditional finned heat sink design. This baseline heat sink is limited to 1 mm fin thickness to remain comparable to the generated designs.

The specific objectives of this thesis are:

- Develop a method to generate and analyze heat sink designs that are not restricted by manufacturing techniques.
- Develop an optimization technique to converge on an ideal design utilizing the unrestricted techniques.
- Analyze and compare the results against traditional finned heat sinks, utilizing two different power bases for confirmation.

1.6 Outline of Remainder of Thesis

The rest of the thesis is organized in the following manner:

- Chapter 2: Literature review covering the design of heat sinks (both typical and advanced designs), the various types of additive manufacturing techniques and material effects and the optimization techniques to find an optimum solution.
- Chapter 3: Methodology and design approach
- Chapter 4: Results and discussion of the analysis
- Chapter 5: Conclusions and recommendations for further work

2. Literature Review

This chapter reviews relevant literature in areas of heat sink design and optimization manufactured using traditional techniques, additive manufacturing technologies that can enable advanced heat sink designs, and the use of genetic algorithms for optimization in general.

2.1. Heat Sink Design and Traditional Optimization

Traditional heat sinks are typically classified as passive or single-phase active systems, with many examples of commercialized systems. More recently, research efforts have focused on more advanced heat sink designs, specifically systems such as two-phase active and microchannel systems. These advanced thermal solutions are known for being highly effective, but a trade-off is made for complexity and cost, which may limit their usage. (Lajevardi *et al*., 2011). To enable comparison with the additive manufactured design techniques proposed here, it is important to understand how these heat sink technologies are currently manufactured and designed.

2.1.1. Traditional Heat Sinks

Lee (1995) conducted a detailed review on both heat sink design and manufacturing. This was discussed previously in Chapter 1, but will be reiterated to provide context. For passive and active air-cooled heat sinks, stamping is a method that forms sheet metal into the fin shape. Typically, heat sinks produced via this method are utilized in low power situations. Another common method is extrusion, which in which fin profiles are continuously formed from a base stock, yielding a heat sink as a single piece. These fins can be effectively utilized in passive or active systems, but may be limited dimensionally by the form factors of the devices they are in. The next method is bonded heat sinks, which have a base manufactured with slots, and then have

fins bonded into the base via thermally conductive epoxy. These heat sinks can be quite effective given their potential for more fins as well as a greater fin to height ratio then that offered from other methods. Cast fins utilize molds and metal injection to create the part. This manufacturing method has typically been utilized for pin fin geometries which would be difficult to form via extrusion or stamping. Finally, folded fins are manufactured by folding sheet metal, and then bonding it to a base. This method is best utilized in situations where bonded or extruded fins may not be practical. Examples of these heat sinks are provided in Figure 2.1.

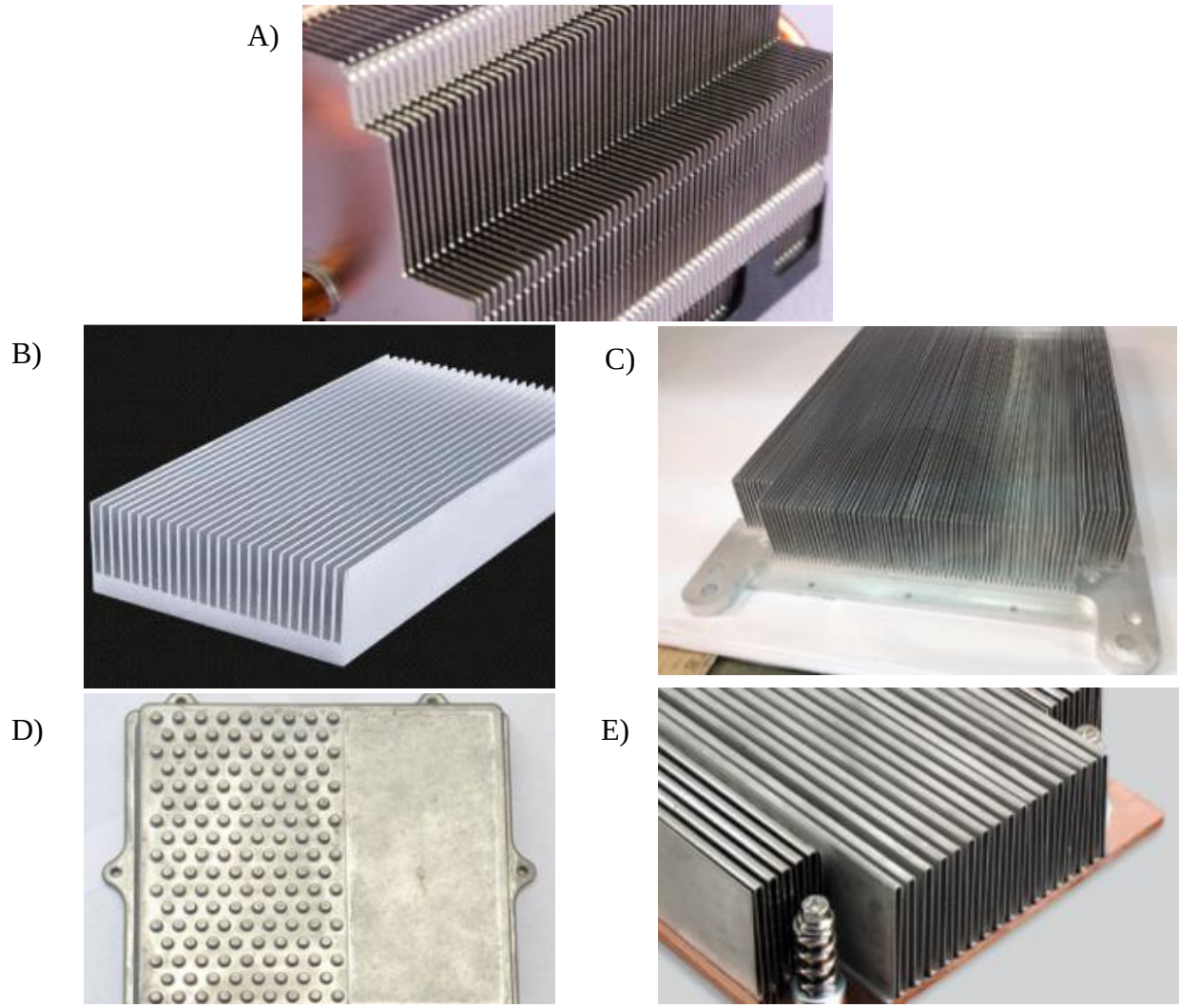

Figure 2.1: Variety of manufactured heat sinks. A) Stamped B) Extruded C) Bonded D) Cast E) Folded (Source Cooling, 2017)

Lee (1995) also discusses the effectiveness of various heat sink designs. In low power applications (on the order of 5 to 50 watts), passive, air-cooled heat sinks may be utilized for dissipation. Passive heat sinks are typically either a heat spreader or an extruded fin base. Assuming the power is manageable, passive systems have the benefit of being both quiet and low cost (Lee states costs of $0.50 in large batches at the publishing time). If more power needs to be rejected, an active system is the next step, with the Lee (1995) suggesting a range of power from 10 to 160 watts for traditional air cooled systems to handle. Active systems utilize a heat sink in addition to a fan or pump to force convective cooling, utilizing either air or a liquid. For air cooled systems, Lee (1995) suggests that the total system will cost approximately $10 to $20 each in a large volume batch. The liquid cooled alternatives can manage far greater power, but at higher cost, with predictions the range of $10 to $100. The costs of various methods compared to their respective performance is shown in Figure 2.2.

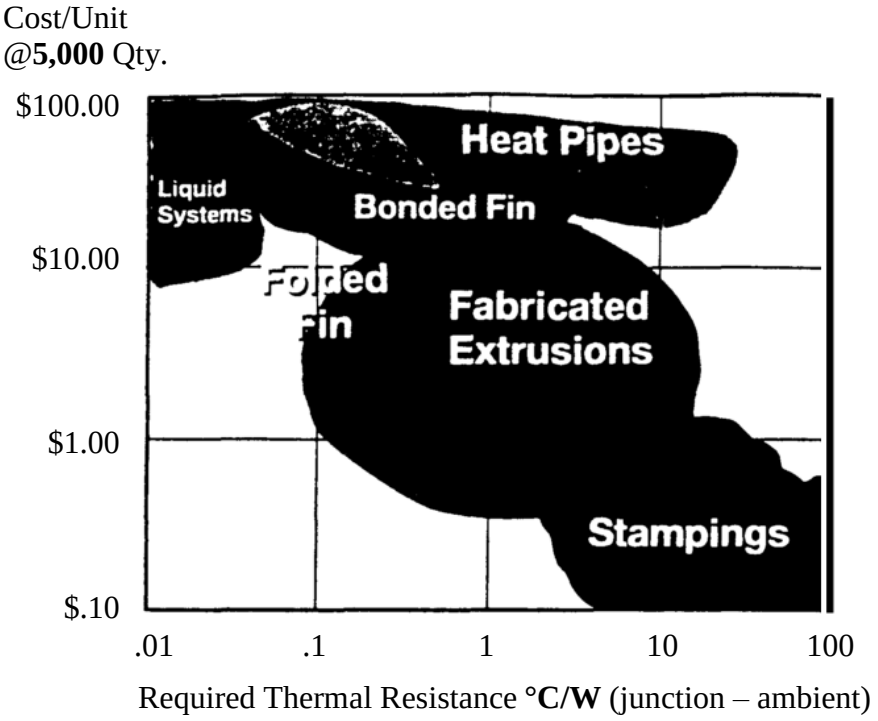

Figure 2.2: A comparison of heat sink technology cost and respective thermal performance by Lee (1995)

Different design and optimization approaches have been undertaken to balance cost, performance, size, and other parameters. Heat sink design and optimization remains an active area of research due to its commercial importance. A few relevant studies are reviewed here to provide an overview of different design techniques.

Ritzer and Lau (1994) focused on a comparison of different systems with a fixed power input, with the goal of minimizing unit cost. Three different designs are examined: a large, low fin density natural convection heat sink, a low fin density forced convection heat sink and a high fin density forced convection heat sink. The fin density of the low density forced convection heat sink was limited to that of the low density natural convection system. Because of the consistent power input and cooling requirements, the size of the heat sinks vary greatly, which factor into the total system costs. It was noted the heat sinks of the low-density passive and active systems utilize the same manufacturing process (extrusion), while the high-density heat sink utilizes bonding. The active system, of course, can utilize a smaller heat sink, exchanging costs with addition of the fan. Based on the final analysis, they find the high-density passive system to be the cheapest per unit, closely followed by the low-density passive and then the active. While it is an interesting result, and demonstrates the effects of fixing system variables, the applicability of this test may be limited in electronics and high-power systems. This is simply because of the restrictive volumes of these systems, and the great power inputs in these small spaces limits the use of large passive heat sinks.

In an effort to see what simple manufacturing additions could be done to improve traditional heat sink performance, Small *et al*. (2006) investigated fin modifications meant to provide greater surface area. A few different designs were collated in the beginning of the analysis, with a mixture and parallel and staggered fin designs. The highest performing design

utilized dimples in the surface of the fins to maximize available surface area. This system was analyzed numerically and experimentally, and compared to a few additional designs. Namely, one that uses a combination of dimples and bumps in the surface. Since it is difficult to describe these shapes, an example of them are displayed in Figure 2.3. In numerical simulation, it was found that the addition of dimples simply resulted in more uniform heat transfer across the channel compared to straight. In staggered fins, dimples actually provided greater heat transfer, but in exchange for a higher pressure drop. Studies were then conducted on the combination dimple and bump system. Results proved even more promising, with this design offering the greatest thermal performance (a 22% reduction over the plain fin) in exchange for a 34% increase in pressure drop with parallel fins. These results were greater than any of the staggered fin alternatives, which did not see any improvement in thermal performance, and maintained a higher pressure drop. Results from the study were quite promising, demonstrating that significant gains may be seen in traditional heat sinks with improvements in the manufacturing process.

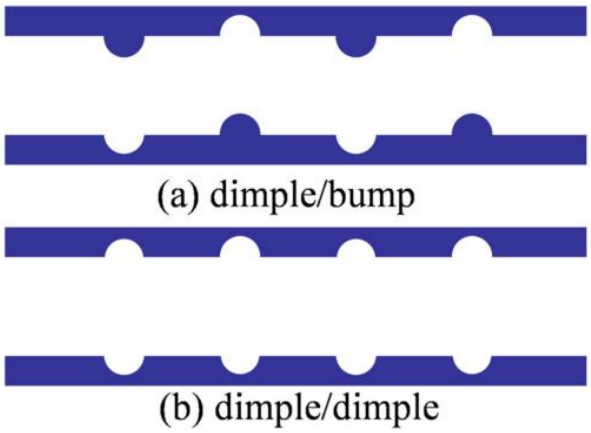

Figure 2.3: Example of the fin geometries investigated by Small *et al.* (2006)

Luo *et al*. (2009) investigated optimization of passive, plate-fin heat sink design. Of particular interest was the methodology for determining the optimum cooling design. A numerical model was developed to determine the capabilities of the heat sink design. This is

done through determination of the total dissipation of the heat sink, including the base and the fins. This requires calculation of heat transfer coefficients of both components, requiring a combination of Grashof, Rayleigh and Nusselt equations based on the individual geometries. With the equation sets established, variables were chosen to create matrices with corresponding solution values. Goals were then set to optimize the result between minimizing total material utilized or by minimizing the heat sink volume, and the solver ran the equation sets and iterated for the heat sink geometry. They were able to achieve effective results, maintaining the temperature of the 112 W lamp below 46 °C. However, it is difficult to say if other optimization techniques may prove more effective over the basic iterative methods, such as entropy minimization techniques.

Stafford *et al*. (2010) studied low profile, active air-coupled heat sink design with interest in small scale powered devices. Images of the heat sinks are shown in Figure 2.4. Their main interest was in comparing the effectiveness of finned and finless heat sink designs with varying height, to determine the point at which a finless design is more effective. The heat sinks were varied in height from 1 mm to 4 mm. As height decreases, the heat sink grows in length. The findings indicate at low fan speeds and the shortest heights, a finless design can provide less thermal resistance than its finned counterpart. While removing fins can decrease cost, pressure drop at small heat sink channel heights can be large. This tradeoff between high heat transfer and pressure drop is continuously encountered in the heat sink industry. This is an important to note for general application, as small channels may cause excessive parasitic power consumption that will not benefit the overall system.

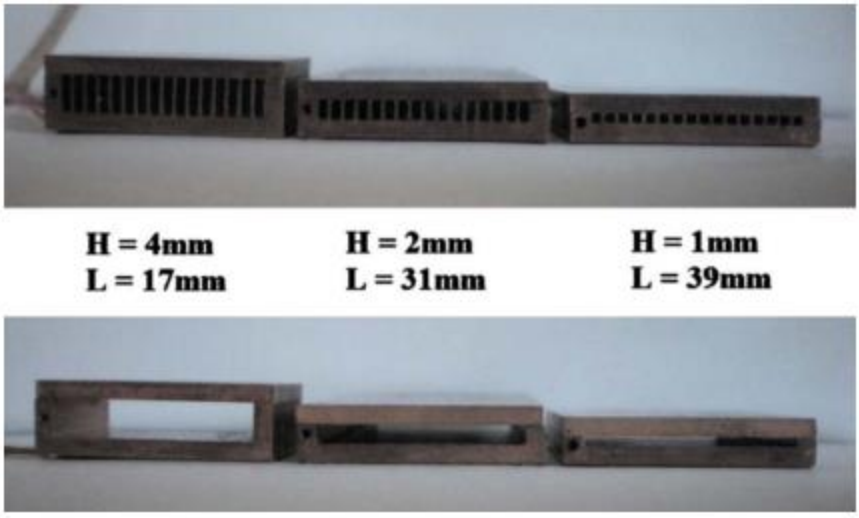

Figure 2.4: Finned and finless test examples from Stafford *et al.* (2010)

2.1.2. Heat Sink Design Using Entropy Generation Minimization Methods

In the above studies, the primary tradeoff in heat sink design was between improved heat transfer and pressure loss, suggesting a multi-objective optimization. In fact, these two parameters can be collapsed into one, using the concept of entropy generation rate. The general entropy minimization method was pioneered by Bejan (1982). An expression for entropy generation rate (2.1) was derived by Bejan (1982), and has been used in several other studies for optimizing heat exchangers (Culham & Muzychka, 2001; Luo *et al.*, 2009).

$$\dot{S}_{gen} = \left(\frac{\dot{Q}_b^2}{T_\infty T_b}\right) R_{th} + \frac{\dot{m}\Delta P}{\rho T_\infty} \tag{2.1}$$

Here, $\dot{Q}$ is the power input (W), m= fluid mass flow rate (kg/s), ΔP= pressure drop (Pa), T_∞ = Volume average temperature of the working fluid (K), T_b = Max temperature of the base (K), and ρ = fluid density (kg/m^3). Starting with the energy balance for an open system (2.2), a few assumptions are made. First, it is assumed the system is steady, meaning time variables can be eliminated. There is no change in velocity or height in the system (i.e., negligible change in

potential or kinetic energy of the system), and the assumption is made that the temperature difference between the inlet and outlet of the fluid is small enough that the enthalpy difference is negligible. This was confirmed by checking the enthalpy and entropy rates over a 5 K increase in a water flow, which resulted in less a 2% change in value for each. Expanding the heat input and work terms provides the power values in equations 2.3 and 2.4. Note that $\dot{Q}_\infty$ represents power input from the fluid.

$$\frac{dE}{dt} = \dot{Q} - \dot{W} + \sum \dot{m}_i(h_i + \frac{1}{2} * \mathrm{V}_i^2 + gz_i) - \sum \dot{m}_o(h_o + \frac{1}{2} * \mathrm{V}_o^2 + gz_o) \tag{2.2}$$

$$\sum \dot{Q} = \dot{Q}_b - \dot{Q}_\infty \tag{2.3}$$

$$\sum \dot{W} = F_D U_\infty \tag{2.4}$$

Similar assumptions are made about the entropy balance (2.5). Steady state and minimal temperature delta between the inlet and outlet of the fluid flow. The heat input and temperature term is expanded to the inputs provided in equation 2.6.

$$\frac{dS}{dt} = \sum \frac{\dot{Q}_i}{T_i} + \dot{m}_{in} s_{in} - \dot{m}_{out} s_{out} + \dot{S}_{gen} \tag{2.5}$$

$$\sum \frac{\dot{Q}_i}{T_i} = \frac{\dot{Q}_\infty}{T_\infty} - \frac{\dot{Q}_b}{T_b} \tag{2.6}$$

The simplification of terms results in the initial equations presented by Bejan (1982) in equations 2.7 and 2.8.

$$\dot{Q}_b - \dot{Q}_\infty + F_D U_\infty = 0 \tag{2.7}$$

$$\dot{S}_{gen} = \frac{\dot{Q}_{\infty}}{\mathrm{T}_{\infty}} - \frac{\dot{Q}_{b}}{T_b} > 0 \tag{2.8}$$

Eq, (2.7) can be substituted into Eq. (2.8), yielding: multiply out the equation by $T_{\infty}T_b$ (2.12) and divide it out afterward (2.13). Combine terms (2.14).

$$\dot{S}_{gen} = \frac{\dot{Q}_{b}}{T_{\infty}} - \frac{\dot{Q}_{b}}{T_b} + \frac{F_d U_{\infty}}{T_{\infty}} \tag{2.9}$$

Then, Eq. (2.9) can be multiplied by the product $T_{\infty}T_b$, and simplified, as shown in Eqs. (2.10) - (2.12).

$$(T_{\infty}T_b)\dot{S}_{gen} = \dot{Q}_b T_b - \dot{Q}_b T_{\infty} + F_d U_{\infty} T_b \tag{2.10}$$

$$\dot{S}_{gen} = (\frac{\dot{Q}_b T_b}{T_{\infty}T_b} - \frac{\dot{Q}_b T_{\infty}}{T_{\infty}T_b}) + \frac{F_d U_{\infty}}{T_{\infty}} \tag{2.11}$$

$$\dot{S}_{gen} = (\frac{\dot{Q}_{b(T_b - T_{\infty})}}{T_{\infty}T_b}) + \frac{F_d U_{\infty}}{T_{\infty}} \tag{2.12}$$

Now the thermal resistance of the system is introduced (2.13). It is rewritten as shown in Eq. (2.14) and substituted into the Eq. (2.12), yielding Eq. (2.15).

$$R_{th} = \frac{T_b - T_{\infty}}{\dot{Q}_b} \tag{2.13}$$

$$T_b - T_{\infty} = R_{th}\dot{Q}_b \tag{2.14}$$

$$\dot{S}_{gen} = (\frac{\dot{Q}_{b}^{2}}{T_{\infty}T_b})R_{th} + \frac{F_d U_{\infty}}{T_{\infty}} \tag{2.15}$$

Now one term of the equation is complete, the other term needs to be rewritten into a form utilizing pressure drop. The drag force is related to the pressure drop multiplied by the frontal area (2.16). The term is rewritten (2.17), and multiplied by the density over the density (2.18). Now relate the mass flow rate (2.19), and substitute it into the equation. The final expression for entropy generation rate is then derived (2.1).

$$F_d = \Delta P A \tag{2.16}$$

$$\frac{F_d U_\infty}{T_\infty} = \frac{\Delta P A U_\infty}{T_\infty} \tag{2.17}$$

$$\frac{F_d U_\infty}{T_\infty} = \frac{(\rho U_\infty A)\Delta P}{\rho T_\infty} \tag{2.18}$$

$$\dot{m} = \rho U_\infty A \tag{2.19}$$

$$\dot{S}_{gen} = \left(\frac{\dot{Q}_b^2}{T_\infty T_b}\right) R_{th} + \frac{\dot{m}\Delta P}{\rho T_\infty} \tag{2.1}$$

Culham and Muzychka (2001) used the Bejan (1982) entropy generation minimization technique to optimize straight finned heat sinks. Equations are based around the thermal resistance of the heat sink fins and base, as well as the associated friction effects for determining force. Parametric optimization was performed to determine the parameters, which include the fin thickness, fin spacing, base thickness, number of fins, device width, device height and device length. To optimize, the device is constrained by a number of variables, and then solved for another variable based off minimizing the entropy generation rate. In the majority of these tests, an ideal solution is determined after 6 iterations, seeing only minor improvements with

additional iterations. Results show the methods are effective, but they caution that restrictions will need to be recognized to maintain feasibility and manufacturability of the device.

One application of the entropy generation rate minimization technique is applied to pin fin heat sinks by Khan *et al*. (2008) The entropy generation rate term used is again similar to that developed by Bejan (1982), with substitutions made for the system thermal resistance and pressure drop. The thermal resistance of the pin fin structure is established through an equation set, and a similar pressure drop model is devised. Analysis is completed through an iterative method, and results were varied based on system variables. These variables included the number of pins, pin diameter, pin height and the fluid velocity. The heat sink was limited in area and height, with a set base thickness and a uniform power applied. Two different materials were compared as well, using a plastic composite alongside aluminum. The results show distinct minimum values in entropy generation rate for each of the dependent variables, meaning there is also a distinct optimum for each of the independent variables. However, it is important to recognize that interaction between variables must be taken into account. In the end they were able to converge on an optimum result between different arrangements and materials, through repeated iteration to minimize entropy.

Finally, Yu *et al*. (2011) investigated passive cooling solutions with radial finned heat sinks. A model was utilized which couple velocity and pressure in the system (verified with experimental data), based off the entropy minimization method. Their model considered three different fin layouts: long fins only, long and medium fins, and a combination of long, medium and short fins. The long fins varied from 3.5 to 5.5 cm, medium fins from 0.5 cm to 0.45 cm, while the sizing of the small fins are unfortunately not discussed. From figures provided, it appears the small fins may be approximately 1/3 the size of the medium fins. Findings suggested

that with an increasing number of fins, the highest temperature region of the heat sink shifts from the center toward the outer edges. This results in less effective heat transfer, potentially counteracting the gains from increasing surface area with additional fins. Analysis showed that the most effective heat sink design was a combination of the long and medium fin lengths, indicating an appropriate balance of surface area and proper material location is important to the system effectiveness. Due to dissimilarities in the problem and reported results, it is unfortunately difficult to compare directly with Luo *et al*. (2009)

2.1.3. Advanced Heat Sinks

The need to remove more power over a smaller area has spurred interest in active phase (two-phase) change and microchannel heat sinks. Phase change heat sinks have significant additional complexity due to the more complicated boiling physics, the need to design the system to prevent burnout (which causes significant damage), and requires additional equipment (such as condenser) to turn the gas back into a fluid, and extract heat. (Ghiaasiaan, 2008) This all comes at a cost, with Lee (1995) estimating the systems to ranging anywhere from $15 to $500. However, heat removal by phase change systems may be on the order of kilowatts, as suggested by Gillot *et al*. (2000). Despite capability, the costs of these system remain significant, indicating usage should be limited to high heat flux systems.

Microchannel heat sinks remain an intriguing technology, which can also be applied in two-phase systems, but are commonly used in single-phase configuration. These microchannel systems take advantage of a reduction in length scales, which allows utilization of significant surface area in a small volume (Paul & Peterson, 1999). This scaling also means that microchannel systems can effectively be utilized to manage high heat flux systems. Research by Kandlikar (2005) suggests a single-phase system that is capable of managing 800 W of power

dissipation, similar to that expected of full scale two-phase flow systems. Again, this capability comes at a price. Lajevardi *et al*. (2011) suggests that in large batches of 10,000 units per year, costs of microchannel design could be minimized $500 per unit. Note, that smaller batches will increase costs significantly, with the suggestion that a batch of 200 heat exchangers per year would cost $5000 each.

Microchannel systems also may be more difficult to manufacture with the conventional processes discussed above. As an alternative, Paul and Peterson (1999) discuss the process of micro-lamination for the manufacturing of microchannel systems, specifically referred to as Microtechnology-based Energy and Chemical Systems (MECS). They suggest that utilizing manufacturing methods seen in the development of silicon integrated circuits is not adequate for application to these microchannel heat sink devices. This is based on limitations such as limited material choices, being unable to handle high aspect ratios and the associated cost. Instead, they suggest the use of micro-lamination and micromachining with a laser, which is able to manage the issues presented with the silicon manufacturing methods. Channels are machined into material stock using an Nd:YAG laser, adjusting power, pulse frequency and beam diameter to manage cut depth and width. An example of the machined material and a comparison size are provided in Figure 2.5. With the laminae stock cut, it has to be bonded to create the final device. A few different options are presented to accomplish this, including Polyimide sheet adhesive, diffusion soldering and brazing, diffusion bonding and micro projection welding. Polyimide sheets can be placed between surfaces, and bonds are formed when the material is heated and compressed. Diffusion soldering and brazing requires the stock to be plated in the diffusion material, and heat is applied to flow the solder and bond the stock. Diffusion bonding is accomplished by stacking the stock, and applying high temperatures and pressure to the stack.

When done correctly, the material should form strong bonds at the interface, essentially fusing the parts into one. Micro projection welding relies on etching of the material stock. Then, when the laminae are stacked, an electric discharge is applied to the area, welding the material. Interest is placed into diffusion soldering and brazing, because of the low temperatures needed for bonding, and performance once bonded. They tested the procedure on a few microchannel devices, one being a microchannel array. This device was tested for pressure drop across a number of flow rates and compared to a theoretical. They found that laminar flow was well approximated in the device versus initial analysis, but some effects caused the flow rate to be less than predicted in turbulent of flow. A lot of information on the process of manufacturing microchannel devices is provided by the study, and helps to reinforce the estimated cost with the devices.

Figure 2.5: Example of laser machined microchannels provided by Paul and Peterson (1999)

The manufacturing and development costs of mass-produced microchannel heat sinks are further investigated by Lajevardi *et al*. (2011) They discuss their manufacturing process, which includes the initial patterning of the shims, shim bonding, de-paneling/ singulating and

interconnect. The devices are designed as 6”x6” sections, and are cut from 24”x24” panels, meaning 16 devices can be created from each panel stack. A final device took 150 shims at 250 μm thick and had a total bonding area of 400 cm^2 per shim. The cost of goods sold shows that each microchannel device cost $5000 for a demand of 200 devices/year, and reduced to $500 each when demand increased to 10,000-20,000 devices/year. Beyond the point of 10,000 units/year, the cost benefits drop significantly. The reason for this, investment for production equipment increases drastically as the number of units rise because of the need to repurchase tools. Capital equipment costs typically remain around $2,000,000 for lower unit costs, and increases to approximately $2,700,000 around 10,000 units. Beyond 10,000 units, it increases to approximately $4,700,000. While not directly advocating the usage of microchannel devices, this study does effectively display the costs associated with their manufacture.

An interesting approach to coupling microchannel heat sinks with integrated circuits (IC) are discussed by Tuckerman and Pease (1981), who integrated microchannel heat sinks directly into the silicon substrate of the IC. Analysis was conducted analytically and experimentally, with three different experimental designs. They found their best design was capable of dissipating up to 790 W/cm^2, a combination of the use of microchannels and high flow rate water. It is argued that this is a significant improvement over traditional heat sink designs, however, at the expense of high pumping power. Furthermore, it is presently limited in commercial applicability due to the costs and complexities of integrating a thermal solution into the IC manufacturing process.

Other advanced options are available outside of the two-phase and microchannel systems, including the previously discussed (Chapter 1) "grown" heat sink developed by Bornoff and Parry (2015). Certainly, the field of heat sink design and optimization has been extremely active over the past 20-30 years. As required heat removal rates increase, the advantage of utilizing

more advanced additive manufacturing techniques to improve heat transfer is of interest. Potential techniques that can be exploited for heat sink manufacture are discussed below.

2.2. Additive Manufacturing

Additive manufacturing has seen a significant surge in interest in recent years. As a result, research interest in how to exploit this improvement for heat transfer devices has increased, as seen with Bornoff and Parry (2015). However, to become viable for usage in commercial design, metal printing has to be utilized. Therefore, it is important to examine the current status of metal printing technology, as well as any impact the processes have on materials properties.

Wong and Hernandez (2012) provide an overview of the state of additive manufacturing via 3D printing of metal. Powder based methods are the most common techniques used in printing, and are utilized in technologies including Selective Laser Sintering (SLS), Electron Beam Melting (EBM), Laser Engineered Net Shaping (LENS) and Prometal. With the exception of LENS, all the systems rely on a base powder bed for fabrication. SLS requires a chamber filled with inert gas to be heated near the melting temperature of the powder. With metals, a binder may be needed, and a laser provides the additional energy input needed to melt the material. The binder is removed post process, often with heating. EBM is similar to SLS, but relies upon a more powerful electron beam to simply melt the material. Prometal also relies upon the powder bed technique, but purely relies on a liquid binder for processing. The machine applies the binder for each layer, and repeats until the final part is complete. Then the part is finished with cleanup of residual powder, and heat treated to harden the material and binder. Finally, LENS is predominantly used for injection of metal into specific locations. The process uses a laser to melt the material, the material is injected and cools down to form the component. In addition, inkjet based systems are expanding to incorporate metal (“XJet 3D,” 2017). In an

inkjet system, a jet applies a photopolymer ink to a surface. A UV light is then used to harden the layer. The process repeats until the part is completed. All of these techniques would be viable for production of a heat sink.

To examine potential differences in material properties from additive manufacturing, Bauer *et al*. (2015) characterize SLM (Selective Laser Melting) produced Haynes® 230. It should be noted that SLM is another common acronym for SLS. A Concept Laser M2 is utilized, which uses a 200 W Nd:YAG laser. The Haynes 230 powder contained particles varying in size from 15 to 45 μm. For processing, they claim to have maintained the machine at the maximum output of 200 W, and varied the hatch distance as well as scan speed. The volume energy density was calculated based on the laser power, scan speed, layer thickness and hatch distance. Density was compared to manufacturer claims utilizing displacement and mass. They found low volume energy density and larger hatch size resulted in a low relative density at approximately 96.8%. With increasing volume energy density and decreasing hatch size, relative density increases to 99.8%. The granular structures can be viewed in Figure 2.6. This result indicates that printed materials are near identical to manufactured material in density for the right combination of fabrication parameters. Tests were also performed for yield strength, Young's modulus, ultimate tensile strength and percent elongation. They built the device in both a horizontal and vertical layering configuration (relative to the device shape), and compared them to wrought and cast devices. The device built in a horizontal configuration generally showed higher strength than the vertical. Printed devices saw much higher yield strength, slightly higher tensile strength and about equal Young's Modulus to the wrought and cast devices. This is with a tradeoff in showing less elongation before breaking. Overall, it seems that printed parts can match cast and wrought pieces in density, while gaining strength and losing ductility. The density of the printed parts is

important to note, as properties of the device would likely be effected if the density was lower and the printed parts had greater porosity. This would probably limit conductivity of the metal as a result of the decrease in conduction boundaries from the additional voids. Unfortunately, a study has not been located where the thermal conductivity of a printed part has been determined.

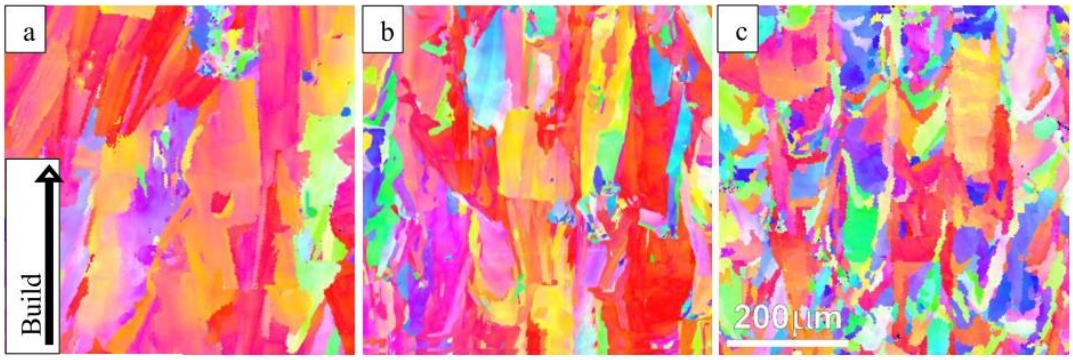

Figure 2.6: Granular microstructure of Haynes® 230 with varying energy inputs. a) 116 J/mm^3, b) 77 J/mm^3 and c) 66 J/mm^3. (Bauer, Dawson, Spierings, & Wegener, 2013)

Murr *et al*. (2012) performed a similar study to Bauer *et al*. (2013) , but did not perform any density studies. Instead, they investigated the micro-indentation and Rockwell hardness tests for a variety of different materials produced from Electron Beam Melting (EBM) and Selective Laser Melting (SLM). The samples were also examined under 3D light optical metallographic and electron microscopy to determine the material structure. Hardness of the printed materials were determined to be generally greater with the SLM produced samples. Similarly, samples built in a horizontal orientation showed greater hardness than vertical built samples for Ti-Al and EBM Inconel 625. Otherwise, the vertically built structures showed higher or equal hardness for Copper, Cobalt-Chromium, SLM Inconel 625, 17-4 Stainless and SLM Inconel 718. With the exception of EBM Inconel 625, the materials roughly matched or exceeded the hardness of their wrought or cast counterparts. The entire list of results is provided in Table 2.1. Between the two papers, the results indicate that material properties of printed materials are close enough to the

cast and wrought parts, that their material properties could be reasonably applied to the additive part.

Table 2.1: Table of material hardness from additive manufacturing processes compared to traditionally manufactured materials. (Murr *et al.*, 2012)

System Fabricated	Z-axis orientation		X–Y orientation		Commercial wrought	Commercial cast
	Horizontal (H)*	Vertical (V)*	Horizontal (H)*	Vertical (V)*		
aTi-6Al-4V: EBM	HV: 4.2 GPa HRC: 45	HV: 3.5 GPa HRC: 42	–	–	HV: 4.1 GPa HRC: 50	HV: 3.5 GPa
Ti-6Al-4V: SLM	HV: 4.3 GPa HRC: 46	HV: 4.0 GPa HRC: 44	–	–	–	–
Cu: EBM	HV: 0.81 GPa	HV: 0.87 GPa	–	–	Anneal HV: 0.57 GPa	–
Co-base alloy: EBM (Co-Cr alloy)	HV: 4.5 GPa HRC: 47	HV: 4.6 GPa HRC: 48	–	–	–	ASTM F-75 Annealed HRC: 25-35**
Inconel 625: EBM	HV: 2.8 GPa HRC: 14	HV: 2.5 GPa HRC: 13	–	–	HRC: 40 Anneal HRC: 20	–
Inconel 625: SLM	HV: 3.9 GPa HRC: 40	HV: 4.1 GPa HRC: 45	HV: 4.0 GPa HRC: 40	HV: 4.4 GPa HRC: 45	–	–
17-4 PH Stainless: SLM	HV: 3.8 GPa H900 HV: 6.1 GPa† HRC: 29 H900 HRC:43†	HV: 3.8 GPa H900 HV: 6.1 GPa† HRC: 29 H900 HRC: 43†	–	–	HRC: 35 H900 HRC: 45†	–
Inconel 718: SLM	HV: 5.6 GPa HRC: 38	HV: 5.8 GPa HRC: 39	HV: 5.4 GPa HRC: 38	HV: 5.6 GPa HRC: 35	Annealed HRC: 24 Aged HRC: 40	–

As discussed in Chapter 1 by Thomas and Gilbert (2014), the costs of additively manufactured parts do not experience a significant change with increasing quantity. This is in contrast to traditional manufacturing techniques, which have economies of scale effects. Meaning that with greater quantities of units, the cost of each decreases. Therefore, it is important to understand the variabilities in manufacturing based on the part produced.

2.3. Genetic and Other Optimization Algorithms

Genetic algorithms were identified as a viable optimization technique to investigate the potential of additivity manufactured heat sinks in the present research. In this section, the genetic algorithm process and techniques are reviewed as well as case studies for their usage.

2.3.1. Genetic Algorithm Techniques and Alternative Methods

A basic overview of genetic algorithm techniques was provided in Chapter 1. A more detailed review was conducted by Gen and Cheng (1999). The first important discussion from their review is on the predominant genetic operators: crossover, mutation and elitism. Crossover utilizes two parents, then randomly takes data from each parent and combines them to create the child. Crossover has the greatest effect at the beginning of generation, as solutions are still stochastic. Near the end of generation, when solutions have started to converge, it will have less effect because of how similar the solutions are. Next is mutation, which randomly changes data in the parent. Behavior is opposite that of crossover, in that mutation has little effect in the beginning, but has greater impact at the end when the solutions have begun to converge. Elitism is simply the best parent solution being carried over unchanged into the new generation. Approaches to multi-objective problems are the main discussion point. Five different approaches on how to assign fitness to a solution are investigated. The first is vector evaluation, in which parent solutions are randomly shuffled before genetic operators are used to create the new generation. It does not explicitly contain fitness evaluations like other approaches. The weighted-sum approach is next, in which the solution's objectives are assigned a fitness weight to determine how optimal the solution is. Essentially, this changes the system to a single-objective optimization problem. Third is the Pareto-based approaches, which offers two different methodologies. The first of these is a ranking based method where different Pareto lines define rankings (i.e. the closer toward the system asymptote the better the ranking), and the second is the tournament method. The tournament method compares candidates to determine if one is dominated by the other, and if not, a sharing method determines a winner based on a niche count (also discussed by Horn *et al*. (1994)). The non-dominated solutions are used to develop the new

generation. Fourth approach is the compromise, which identifies solutions that are closest to an ideal solution as a measure of distance, rather than relying upon a Pareto front. Finally, the goal programming approach sets a target value, then takes a weight sum of deviation of objective functions in each solution for ranking. A number of different approaches can be taken to drive the algorithms, the best choice has to be determined for the specific problem.

Another detailed review on genetic algorithms is provided by Baluja and Caruana (1995). Rather than focusing on the mathematical formulae of the algorithms, the focus is on practical application and study of the effects of various parameters. The discussion starts with the typical parameters utilized and adjusted: population size, crossover type, crossover rate, mutation rate and elitist selection. Population size is self-explanatory, the number of different solutions per generation. Three different crossover types were utilized in the research. One-point crossover chooses a single point in a solution, and randomly chooses a crossover value between two parents. Two-point repeats this with two crossover points. Uniform crossovers compare the parents for each solution point, and randomly picks between point values from the parents. Crossover rate is percentage of time that a crossover occurs, and if it does not occur, the parents pass-through unchanged. Mutation rate is the probability that a solution point will be randomly changed. Finally, elitist selection ensures that the best solution from each generation is carried to the next generation. This helps ensure that each generation increases in fitness, and does not lose the best possible solution to errors. A four-peak optimization problem was selected to provide the test bed for genetic algorithms. It is an important note that this problem was designed to favor one-point selection. Results from the test showed a few tendencies. Among them, elitist selection algorithms performed better than algorithms without the selection, larger population sizes performed better than smaller ones and finally, the single point crossover performed the best, as

was expected. Interestingly, performance did not see significant impact from variations in crossover and mutation rate. The results from this study are shown in Figure 2.7. Moving on, a comparison is provided against an alternative methodology, referred to as Population-Based Incremental Learning (PBIL). PBIL strives to create real value probability vectors, which reveal effective solutions with high probability. They start at a midpoint solution value, and then move away as further sampling occurs. The biggest advantage of PBIL is that it can maintain dissimilar points in the generation, whereas a genetic algorithm will cause the points to converge. Results indicates that for the four-peak problem, the PBIL provided better overall evaluations. In some ways, the usage of an external vector for maintaining information throughout results is similar to the methodology used in micro-genetic algorithms, which is discussed by other in other studies.

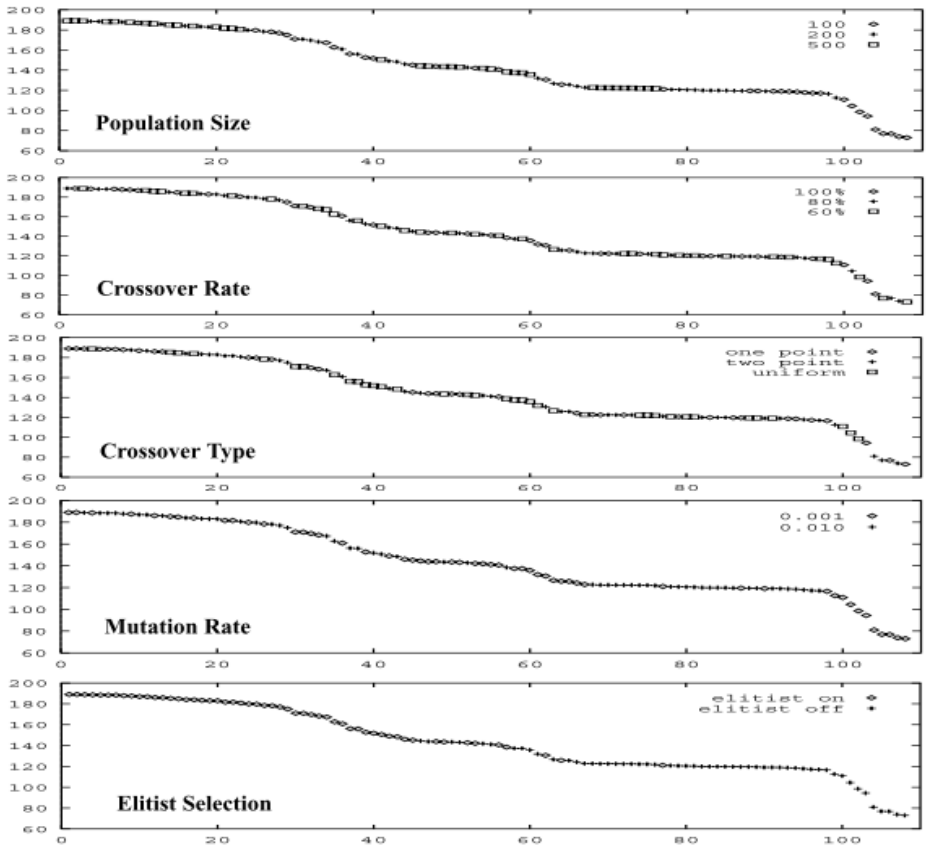

Figure 2.7: Comparison of effects of genetic operator variables on solution results. Note that the X-axis is the test number, and the Y-axis is a performance measure (higher is better). (Baluja & Caruana, 1995)

A good coding guide is provided by Houck *et al*. (2008), who investigated the development of MATLAB code for genetic algorithms. A basic introduction of genetic algorithms is provided, which include six fundamental requirements needed to develop a genetic algorithm code. These six requirements are chromosome representation, a selection function, genetic operators for reproduction functions, creation of an initial population, termination criteria and an evaluation function. Basic parameters for their optimization functions are discussed. Among mutation operators, there are binary, uniform, non-uniform, multi-non-uniform, and boundary operators. Binary, flips each bit in the individual based on probability. Uniform mutation randomly selects a variable, and sets it to a uniform random number, while non-uniform mutation sets a variable to a non-uniform (i.e. varying depending on the probability) number. A boundary mutation selects a variable and sets it to either an upper or lower bound. Finally, the multi-non-uniform crossover applies a non-uniform operator to all of the variables. Crossover operations mentioned are listed as simple, arithmetic, heuristic, and normalized geometric operators. Simple is suggested as a crossover that generates a random number dependent on the parent. Arithmetic creates two linear combinations of the parents that are complimentary. Heuristic is a simple linear extrapolation of the individuals, dependent upon fitness information. An example of this methodology is provided by testing on functions generated by Corana *et al*. (1989). Optimization evaluations were done through two different methods, one determined from returning a value at a point determined by a genetic string and the other through the Sequential Quadratic Programming (SQP) operator in MATLAB. The methods are used with both floating and binary genetic algorithms, finding that sequential quadratic programming greatly increased speed of optimization. The paper is rounded out with a discussion of the MATLAB code that was developed. Unfortunately, they do not discuss the

effects of utilizing the different mutation and crossover functions, despite providing an otherwise useful guide on coding algorithms.

Deb and Deb (2014) discuss a number of different mutation operators for use in genetic algorithms. Among these mutation operators are polynomial and Gaussian methods. Polynomial relies on polynomial functions that relate the initial parent to the mutated through functions dependent on the value of the randomly generated number used to determine mutation. Gaussian mutation changes a variable value to a neighboring one given a probability function. A few different functions are utilized to determine the final value of the mutated result. The operators are tested utilizing a few different schemes. The first being standard mutation, with a predefined probability. Second is a mutation clock, which allows mutation to occur in a set number of variables. The third limits mutation to one per solution with a random variable. Fourth is a fixed strategy mutation, with every variable being given an equal chance, and a set mutation order in the population. Finally, the fifth is a diversity-based mutation, which varies the mutation probability based on the number of variables in the system. It was found the polynomial and Gaussian operators perform similarly, and that the mutation clock operator was quite effective, proving to be both the fastest executing and offering better performance. This paper does provide a variety of potential tools to use in genetic algorithms, but is difficult to implement in a bit-array solution set.

Coello Coello and Pulido (2001) discuss the design of micro-genetic algorithms and compare them to traditional genetic algorithms with the hopes of improving efficiency. To start, two memories are created; a population memory which provides a diversity source, and an external memory to archive the solution set. The population memory is also split into a replaceable and non-replaceable set. An initial population is generated, and used to create the

external memory. Each micro population utilizes crossover, mutation and elitism to generate a new population. The replaceable part of the micro-population is compared to a randomly selected member of the external memory. If the replaceable member dominates the match, they replace the solution in the external memory. If the member of the external memory dominates, it replaces the solution in the replaceable memory. The algorithm flow is displayed in Figure 2.8. An adaptive grid is also utilized to maintain population diversity across the Pareto front, which is not necessary when performing a single objective optimization problem. The algorithm is tested through a few different functions, and compared to typical algorithms. They found they were able to achieve the same results with the micro genetic algorithm as the other algorithms, but with lower computational cost. The techniques provided serve as good basis for the design of a micro genetic algorithm.

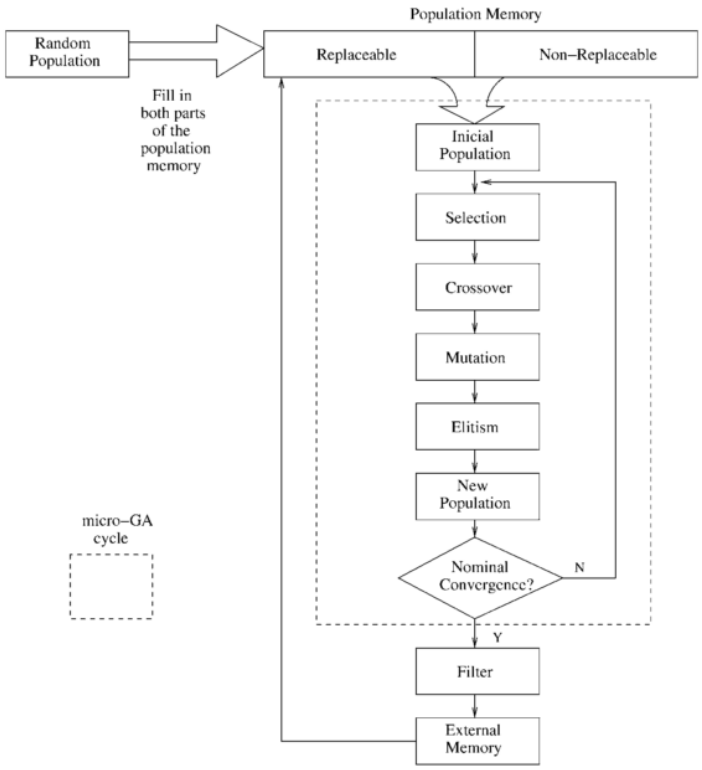

Figure 2.8: Design of a micro-genetic algorithm as defined by Coello Coello and Pulido (2001)

Traditional multi-objective algorithms will usually make use of Pareto fronts to guide optimization. Horn *et al*. (1994) demonstrate the usage of Pareto dominant solutions as a methodology to solving multi-objective genetic algorithms and the technique around it. The basis of these techniques is selection. One of the most widely adopted methods is that of tournament selection, in which the best solution in a set of individuals is moved into the next population. Control can be exerted on convergence speed by managing the size of the tournament, but this method is more ideal for single-objective problems. For multi-objective problems, the Pareto domination tournament is suggested, in which two individuals are compared against a larger individual set. If one of the individuals is dominated by the set, the other is selected for reproduction. If neither or both dominate the set, a winner has to be chosen by shared fitness. In shared fitness, the individual's objective fitness is divided by a niche count to determine the shared fitness value. The niche count is determined by the number of other individuals around it. The resulting method is tested with three different problems, in which are found promising results. However, they are quick to note the importance of an appropriately large population size, as well as having an appropriate comparison set size. If unable to simplify a problem away from a single-objective problem, the Pareto tournament selection described would be the ideal choice.

Bit-array algorithms allow the discretization of geometry into pixels that can be readily manipulated, making them ideal for problems involving a range of potential shapes, as in this research. One of the earlier papers exploring bit-array genetic algorithms is by Kane and Schoenauer (1996). They investigate using a discretized (2x1 aspect ratio) bit-array space to solve a structural loading optimization problem. As with most genetic algorithm problems, the solutions are evaluated and selected before being operated upon by crossovers and mutation. Most useful, a discussion is provided on methods of manipulating bit array systems. With

crossover, variety of techniques are mentioned, including 1-point standard, 2-point standard, diagonal and 2-block. 1-point and 2-point standards can only exchange data between parents across horizontal bands (a result of originating from one dimensional solutions), which leads to geometrical biasing. An alternative is the diagonal exchange, which can exchange information across any straight line in the array. The block crossover splits the domain with two horizontal and two vertical lines, to exchange data across these blocks. An experimental comparison was performed with these operators and compared with a uniform crossover. They find that one dimensional operators are readily outperformed by the uniform and two-dimensional operators. The uniform operator struggles in the early operations as a result of disrupting emerging schema, but converges to the result in later generations. Little comment was made about the two-dimensional operators. Mutation operators have less variability than the crossover methods. A traditional bit-flip mutation is presented, as well as population based and boundary based operators. The population based operator flips a bit based on how many times a certain value appears in that position within the population. If one bit is prevalent, the probability is higher. Boundary based mutation places mutation probability higher at the boundary edges of bits in the array. From trials, population based mutation showed only slightly better performance from the traditional bit-flip method. They reserved boundary mutation as an end of run technique for fine tuning. The structural analysis algorithm was performed and compared against an adaptive scheme. It was found that the algorithm produced results that were similar to the other schema, but with some variation in the shape topology, as shown in Figure 2.9. The algorithm was also expanded to multi-load scenarios on a bicycle, which found similar results. This paper provides a significant amount of guidance that is useful for the development of a bit-array genetic algorithm.

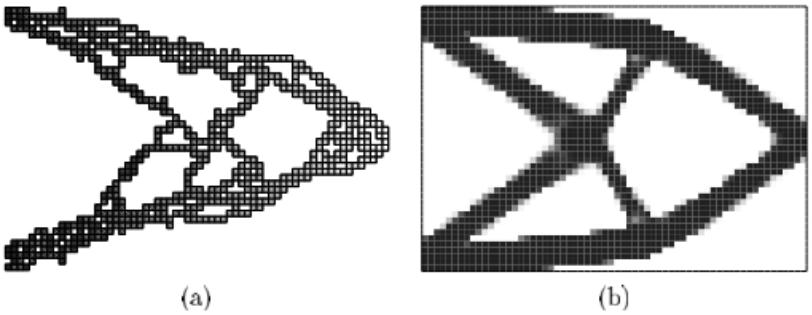

Figure 2.9: Solution of a loading problem solved by (a) the GA-based method, (b) homogenization method. (Kane & Schoenauer, 1996)

An interesting approach to the bit-array generation problem was presented by Pham *et al*. (2014) Rather than treating the bits as either existing or not, the bits are treated as weighted results, ranging from 0 to 1 and modifying the respective material density. To test this process, 3 different scenarios are analyzed with varying loads and fixed location points. The algorithm optimizes the systems for strength, performing structural analysis by combining equations and algorithm operations into a single MATLAB code. A consistent truss design throughout the analysis is reported, with some of edge material being of a varied density. While an interesting technique, it is difficult to apply to variable material properties in most software packages, limiting usage.

One of the alternatives available to genetic algorithms is particle swarm optimization (PSO). A comparison is provided by Eberhart and Shi (1998), starting with the operation of PSO. PSO starts with a population member represented by a particle, which is treated as a point in dimensional space. The particle knows its previous location with the best fitness value, and has a rate of positional change (or a velocity). Weight is added to the system to add inertia, which is taken into account by velocity and position equations. While the PSO does not explicitly contain crossovers and mutations, the effects of the swarm create a similar behavior seen in the particles. Unlike a GA, PSO does not depend on the concept of selection, and has no “survival of the

fittest" operators. While an interesting alternative to a genetic algorithm, PSO does not have the research or optimization strategies that allow it to be effective for utilization with CFD.

Two different optimization studies are performed by Zielinski and Luar (2006; 2006) across two different papers, both offering perspectives on different techniques of evolutionary algorithms. The first paper investigates differential evolution, which utilizes vector based differential equations and compares to a target vector, with special operations if the equation encounters a constraint. Generational changes are managed by mutations and weight crossovers. The variable with the smallest objective function (the best result) is used in the next generation and repeated until optimized. This approach was utilized on 24 different optimization problems, and found it was able to converge on the best solution in two-thirds of all the operation runs. The second paper investigates the usage of particle swarm optimization (PSO), which is based on the idea of social group behavior for optimization. In PSO, each solution has a known position, velocity and personal best position (the settings with the best functional values for the solution). The velocity is constantly changing between each iteration based on the previous velocity, the known results of the group and interaction within the group. Unlike the differential evolution, convergence only failed in three of the test problems. However, a trade-off is made in computation effort, with particle swarm taking more than twice the time to compute of the differential evolution solution. While these methods offer effective tools for different evolutionary optimization problems, it was decided to be a poor choice to apply to a bit-array optimization problem.

There are a few methods available for multi-objective, combinatorial problems. Pareto simulated annealing is discussed by Czyzak and Jaszkiewicz (1998), and start by discussing the difficulty of the method. Two factors are listed as defining the difficulty. First, Multi-Objective

Combinatorial Optimization (MOCO) problems require strong cooperation with the decision maker (DM), meaning high requirements for tools to be effective. The second factor is not well described, simply stating that multi-objective problems were more difficult than single-objective problems. To provide background, the available tools of the MOCO into are classified into the categories of: exact, specialized heuristic and metaheuristic procedures. A satisfactory explanation of each methodology is not provided, only limitations of each are discussed. It is assumed that exact algorithms have some perfect mathematical relationship to the problem, and it is argued that these have high computational complexity and are limited in use. Specialized heuristic algorithms search the solution spaces of specific problems, and are again described as being of limited use. The metaheuristic algorithms are then discussed, and are reported as being more efficient than the other procedures. This is because they allow finding a near optimal solution in a wide space with a comparatively short amount of time. In contrast to the other methods, the limitations of the methodologies are not discussed. It was decided to use Pareto simulated annealing (PSA) for the solution method. Pareto simulated annealing borrows some basic ideas from simulated annealing, such as; neighborhood solutions, acceptance of new solutions with some set probability, dependence of probability on some parameter and schema of these parameter changes. The largest difference with PSA is that it utilizes the population of solutions each iteration along with objective weights, similar to genetic algorithms. Based on the paper's description, it seems reasonable to consider PSA to be a cross of simulated annealing and genetic algorithms. Evaluation determined the methodology to be effective at finding a solution near the most efficient solution set. It is mentioned the number of iterations needed depends upon the population size, as it was established around a set number of solutions. While an interesting

approach, it is not clear how effectively the methodology could be applied to a bit array problem rather than a mathematical one.

2.3.2. Genetic Algorithms for Optimization of Thermal Fluid Devices

The optimization techniques described above can be broadly applied to any problem. Optimization of thermal and fluid systems have particular characteristics that are of interest to the present study. Hilbert *et al*. (2006) investigate the possibility of optimizing a heat exchanger by combining CFD and genetic algorithms. The main parameters of interest are the average temperature difference between the fins and the fluid, and the pressure drop across the array. A multi-objective genetic algorithm is utilized for optimization, using a combination of elitism, crossover, averaging and mutation. Non-dominated Pareto front techniques guide the solutions. The optimization code is developed in OPAL, an optimization package. Geometry and meshing is handled through Gambit, and then CFD performed through Fluent. Changes were made to the spline of the pin/blade geometry, with a set degree of curves based on a polynomial function. When a new generation is created, they are solved analytically in parallel, by distributing the workloads to various workstations. With population sizes of 30 and 20 generations, the combined computing makes the generation significantly faster than if managed by one workstation. Testing showed the system able to generate an effective, converged, Pareto front, with six individual blade profiles presented as an optimum. Many of the methodologies used are similar to the techniques applied in the present study. The addition of bit-array analysis, however, has a significant impact on the analysis methods and speed.

Another application of genetic algorithms and CFD is applied to microchannel heat exchangers by Foli *et al*. (2006) The microchannel is of finned design, with alternating passages of hot and cold fluids. Variables are set as the channel height, channel length, width of the hot

channel, width of the fin and width of the cold channel. The mathematical model used by the CFD software was discussed, utilizing steady state momentum and energy equations. Before delving into the advanced method with genetic algorithms, two different initial studies are conducted based upon design constraints. The first scenario fixes the system volume, and seeks to determine the optimum aspect ratio between the channel height and width. A quick numerical study is done to determine pressure drop, heat flux and heat transfer. The ideal aspect ratio is found to range from 8 to 20, giving a balance between all variables. The second analysis allows the system volume to vary, but restricts the dimensions of the microchannel to a defined range. Using this method, the optimal aspect ratio was determined to be 28, providing a balance of heat transfer and pressure drop. Based on these results, optimization steps are established based how the system is defined and the resulting optimum aspect ratios. With this established, the genetic algorithms are discussed, in which a traditional algorithm is applied for optimization. Crossover and mutation is used, with no mention of an elitism. For the purposes of the algorithm, the height and length of the channels are fixed, as well as the flow cross-sectional area. Results show a large number of data points (shown in Figure 2.10), the best of which form an effective Pareto front, which proved to yield greater heat fluxes than that achieved with the simpler methods. Methods used in the paper are fairly conventional, but the greater distribution of data points compared that seen with Hilbert *et al*. (2006) is a more likely result to achieve.

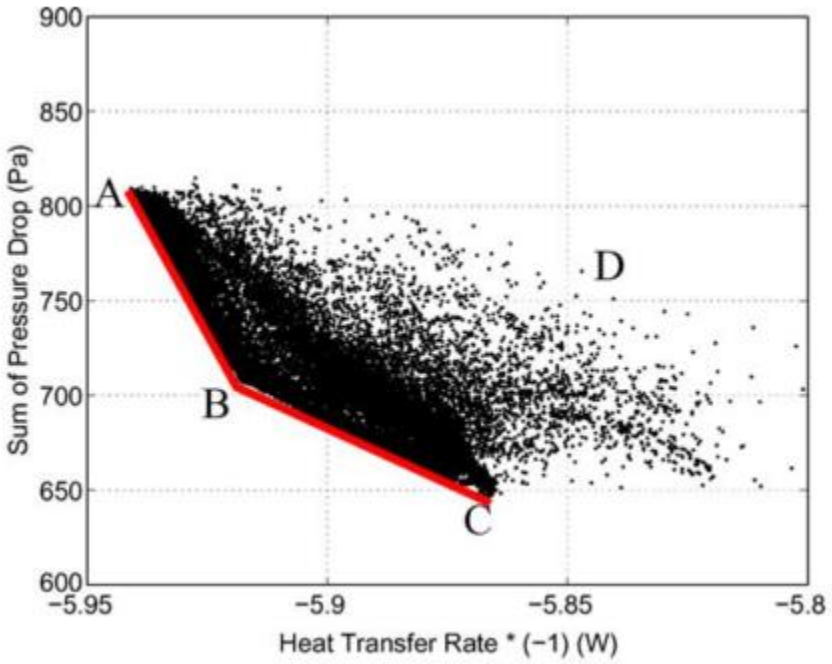

Figure 2.10: Distribution of results from microchannel optimizations by Foli *et al.* (2006)

The viability and performance of stacking microchannel heat exchangers are discussed by Wei and Joshi (2002), and optimized through the use of algorithms. The basic design is simple, utilizing standard finned microchannels, and stacking them vertically. Modeling is done by treating the system as a thermal resistance network, and utilizing an algorithm akin to the Pareto swarm optimization methods. The algorithm starts with a feasible point that meets problem criterion, and generates additional points based on explicit constraints. If a geometric or implicit constraint is violated, the point is moved toward the centroid. After this, if the point now breaks an explicit constraint, it is removed. If all points meet all constraints, it can be evaluated and then checked to see if the functions have seen improvement over the past five moves. If no improvement, function is considered converged, if not the points are moved toward the best solution and recalculated. The process repeats using a different random “seed”. An optimal solution was achieved within 100 steps, and determined the lowest thermal resistances were achieved with layer stacks of 3. Similarly, the optimum aspect ratios (of height to width) and

channel length were determined. Assuming the optimum number of layers, the ideal aspect ratio is approximately 5, and an ideal channel length of 5 mm. This meshes well with the recommendation that stacking short channels is the optimum technique for reduction of thermal resistance.

Szollos *et al*. (2008) also perform CFD analysis, but on airfoils with the usage of micro-genetic algorithms. They explore eta-dominance micro-genetic algorithms and compared to commonly a used genetic algorithm. A few sample functions were used to compare the Pareto fronts (before starting the CFD), and found that for each case, the micro-genetic algorithm matched or outperformed the traditional algorithm. The micro-genetic algorithm was than applied to an airfoil optimization problem, trying to minimize the drag coefficient for a variety of lift and drag conditions. The algorithm was set with a population size of 4, full crossover probability, no mutation, and some range adaptation. Again, they were able to achieve an optimum Pareto solution with the micro-genetic algorithm, while failing to achieve the solution while using traditional algorithm. While trying to show the benefits of the micro-genetic algorithm, it does require consideration if the problems were chosen explicitly to demonstrate superiority of the eta-dominance algorithm. The study was also not explicit in the decision to exclude mutation. It is mentioned they were unsure of its effects on the micro-population, but they also found some problems where mutation helped the convergence. Regardless, the results did show that micro-genetic algorithms can be used effectively with CFD based problems.

One of the examples of a micro-genetic algorithm being applied to a CFD problem is presented by Park (2010), who attempt to optimize the geometry of combustion chambers. The goal of the research was to minimize the fuel consumption while maintaining stoichiometry by altering only the chamber geometry. Geometry is defined by four points, and completed by using

two Bezier curves. Combustion, soot formation and fuel spray models are integrated into the CFD analysis. The algorithm uses a population of 9, uses an elite solution and performs tournament selection to determine the solutions that are utilized for crossover. It does not seem to utilize the external memory typically seen with the micro-genetic algorithms, rather it just maintains knowledge of the best solution through the generation (which is simple elitism). Despite the differences, the results were positive. The system was near optimized within 6 generations, with small gains seen through the remaining 74 generations, as shown in Figure 2.11. Use of elitism meant that the minimum value obtained always remained the same or decreased over generation time. This effectively shows that micro-genetic algorithms can effectively be used to optimize CFD models, with good results.

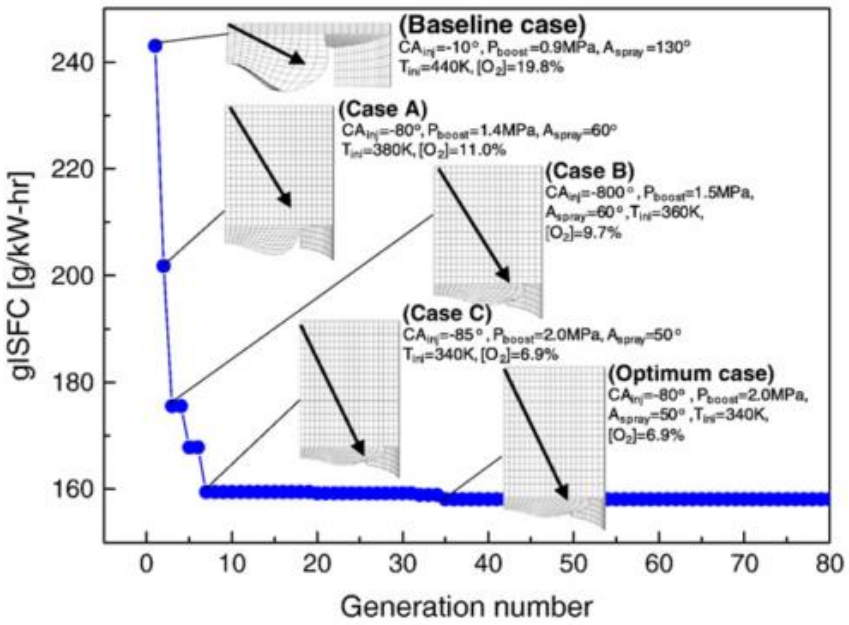

Figure 2.11: Effects of optimization algorithm on shape over generations of designs. Y-axis is gross indicated fuel consumption. (Park, 2010)

Finally, Safikhani *et al*. (2011) investigate the use of CFD with in combination with genetic algorithms for optimization of a centrifugal pump blade geometry. The geometry was

controlled by adjusting angles in the camber curve, with select options for each angle. First, efficiency was calculated for a variety of different flow rates and inter-blade geometry. Second, the geometry was manipulated to try and determine the Pareto front of efficiency and Net Positive Suction Head required (NPSHr). CFD was used until results reached the optimum achieved through case study. An interesting note, is that the CFD solutions show a wide range of results before achieving the final Pareto function, indicating that a large number of analyses need to be run to converge on the solutions.

2.4 Summary and Justification for Current Study

While there is an abundance of research available on heat sink optimization, additive manufacturing and genetic algorithms, there is limited research on applying all three into a singular problem. The closest research has been performed by Bornoff and Parry (2015) and Wu *et al.* (2016), who investigated a grown heat sink intended for production from AM, and examined the use of genetic algorithms to optimize flow paths in a heat sink block. Compared to Bornoff and Parry (2015), the research in this thesis takes an alternative approach to solve a similar problem. While Bornoff and Parry designed a heat sink by organically growing the geometry based on heat flux and iterative analysis, the present study aims to optimize heat sink geometry through the use heat flux and bit-array genetic algorithm optimization. This investigation to explore the feasibility of optimizing the frontal geometry of heat sinks fabricated with additive manufacturing technology using genetic algorithms has not been previously reported in the open literature.

3. Heat Sink Flow Cross Section Optimization

As shown in Figure 3.1, there are a few potential heat sink planes that could be used for optimization. One is the X-Y plane, which is the frontal cross section flow area of the heat sink. The other option would be modifying in the X-Z plane, creating unique flow channels. Simultaneous 3D optimization in the X-Y and X-Z plane is the ultimate goal. However, this creates a significant challenge in a few different manners. First, the generation algorithms become significantly more complex, having to account for both the X-Y and X-Z planes. But more importantly, the number of cells or "voxels" in a 3D space creates a large number of variables in the system. This, combined with the variety of surfaces that have to meshed and analyzed, means that the computational effort is intense, and unrealistic to perform with the computational resources available for the work in this thesis. Thus, modifications only in the X-Y plane were considered in this first step in exploring heat sink optimization techniques. Optimizing the X-Y plane is consistent with traditional finned heat sinks, which do not have feature variation within the X-Z plane.

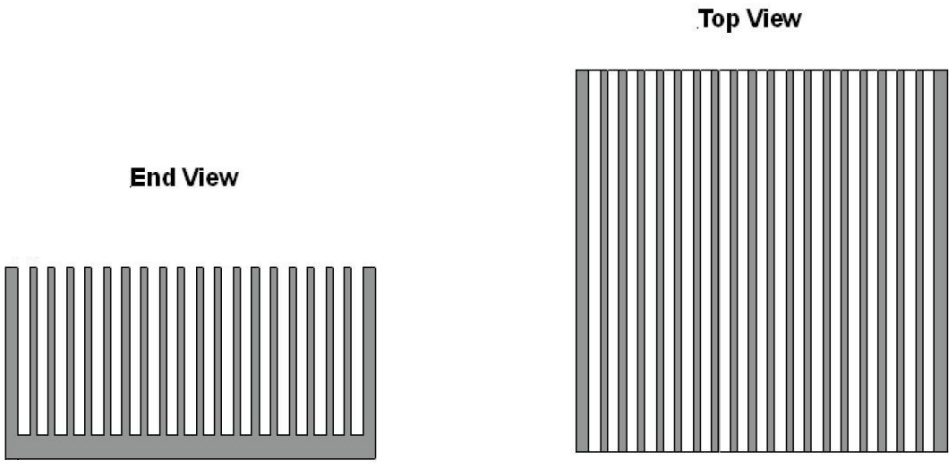

Figure 3.1: On the left, the frontal view representing the X-Y plane. The right shows the top view representing the X-Z plane. (Haskell & Quesnel, 2017)

3.1. Baseline System Assumptions

The base device dimensions are set at 52 mm × 50 mm × 12 mm, with an internal fluid volume of 50 mm × 50 mm × 10 mm. These dimensions were chosen to match with a commercial heat sink (Remsburg, 2007). At each end of the device, a 1 mm thick shroud is extended around the edge, leaving a flow channel the size of the frontal flow area. The shroud extends 5 mm from each end, creating a total device dimension of 52 mm × 60 mm × 12 mm in the simulation. This is shown in Figure 3.2. Grid spacing is 1 mm, with a similar fin width when generated. The grid utilized is shown in Figure 3.3.

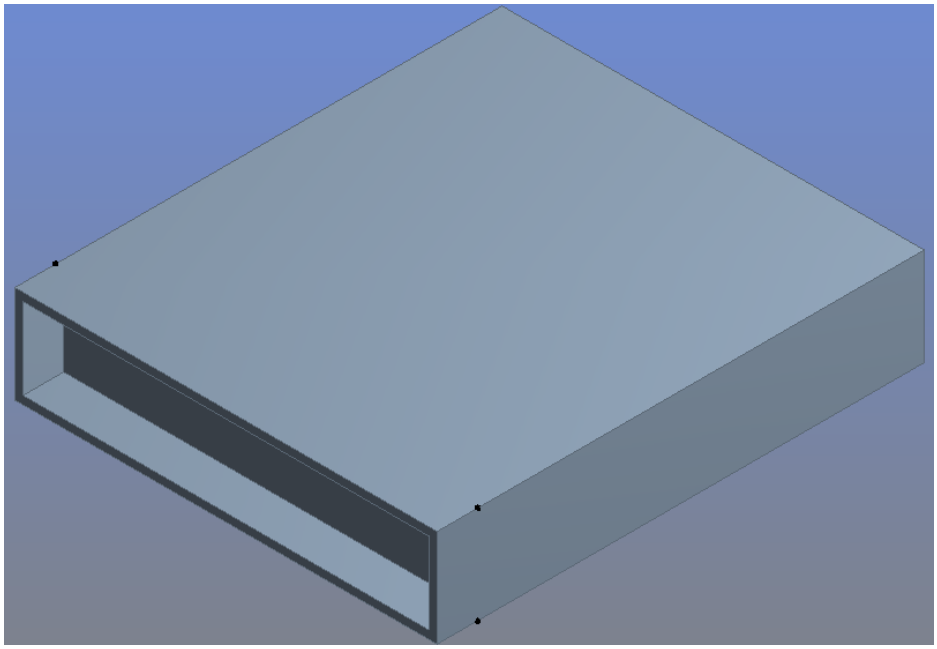

Figure 3.2: The base of the heat sink before the generated geometry is removed.

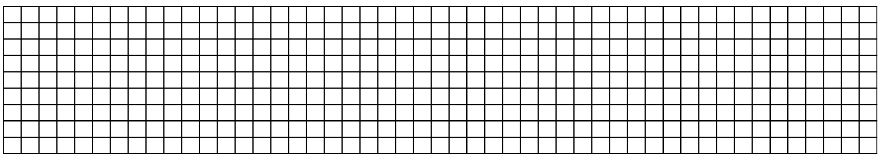

Figure 3.3: Example of the discretized domain used for geometry generation.

The initial fluid temperature is set at 300 K. The fluid used is water, with a low velocity flow of 0.1 m/s at inlet of the frontal cross section. The flow rate is low to minimize pressure drop and pumping power, while also maintain the fluid below the boiling point. It was also intended keep the Reynolds number low to simplify simulations. However, during geometry optimization some of the larger internal flow paths resulted in Reynolds numbers that were partially into the transition regime for circular flow in a duct, with Reynolds numbers around 2000. Finally, the heat sink material used is aluminum. Coding for generation of solutions and power maps were performed in MATLAB (2015).

3.2. Initial Geometry Generation

A power map is required to initialize the problem. The first analysis utilizes a simple symmetrical power map, with a total area of 50 mm × 50 mm. The center 10 mm^2 is set as the hottest area, with a power of 60 W. The second area is 30 mm^2 (minus the center), and has a power of 40 W. The remaining area is 20 W. The second analysis utilizes a power map that is based on a power distribution from a representative non-symmetrical power map, with a maximum power set at 20 W, and a minimum of 0 W. The total power was 700.1 W for the symmetric and 218.7 W for the non-symmetric power maps. Both power profiles are shown in the Figure 3.4 (symmetric) and Figure 3.5 (non-symmetric).

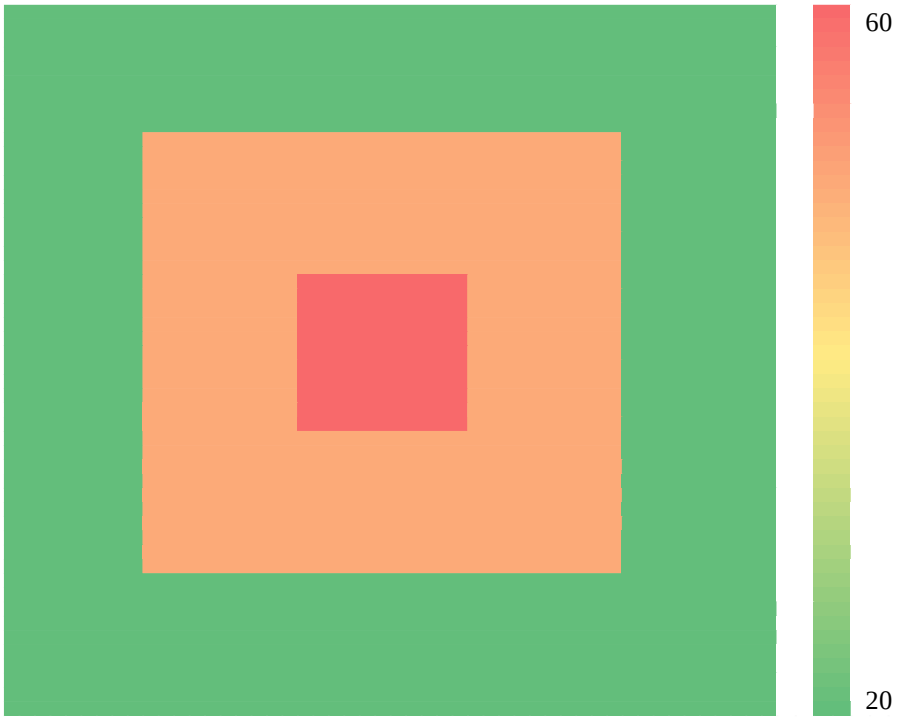

Figure 3.4: Symmetric power map used in the first analysis, units are in W

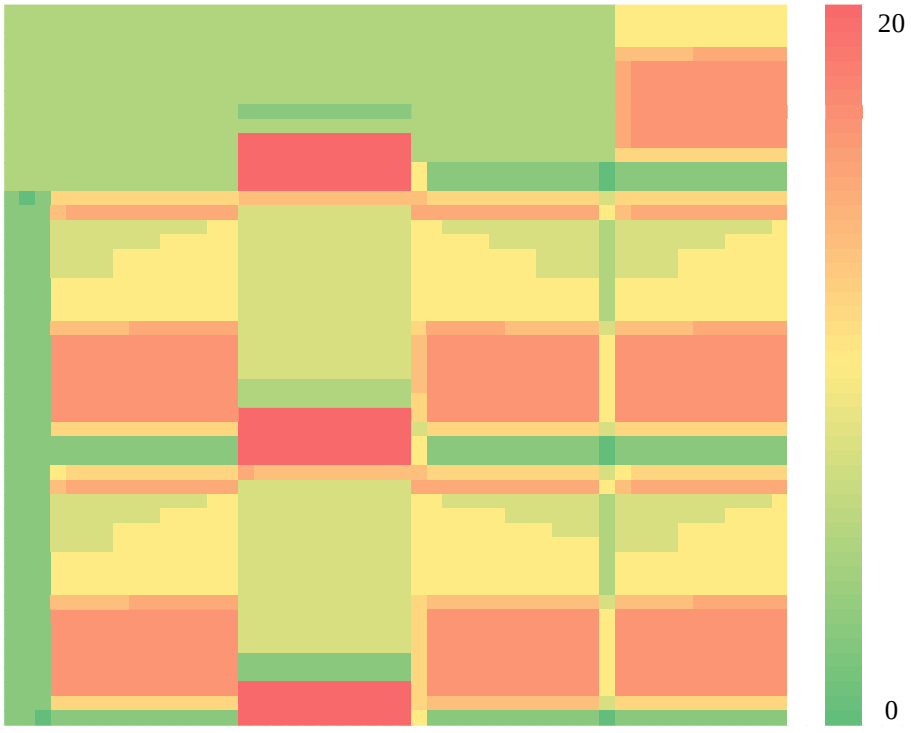

Figure 3.5: Non-symmetric power map utilized in the second analysis, units are in W

Both flux profiles are converted into a CSV file that can be imported into ANSYS Fluent (2016). This file contains coordinate information for individual points in across the heat flux area, as well as information on the power in W. These points represent a discretized location across the base of the device, with associated information. The CFD software has a hard limit of 1412 points for input, which means this processing is necessary to successfully convert the data. The point limit is mostly arbitrary, likely just providing a maximum memory limit in the system. Post processing has to be done to ensure an appropriate array size. This can either be done by strategically selecting points to delete around the area, or choosing to interpolate to a grid size of 37 × 37. Each point in the system is assigned a Y-dimensional value of -0.0011 mm, which is important to accurately place the location of the power map in the 3D space of the solver.

Since optimization in a single X-Y plane is considered, the X-Z location with the highest power output for each power profile was used as a worst-case condition. This is determined by summing the power output in each row and determining the largest value. Once created, the initial 2D geometry generation can begin. The solution space is initialized for the defined geometry space, creating a 49 × 9 grid of points. This mesh size was chosen to balance computational time and convergence reliability. A 50 x 10 grid (what would be expected) was tested, but was found to be highly unreliable in generating solutions that would readily mesh and analyze, failing with over half the generated solutions. The 49 x 9 grid would fail approximately once every ten solutions, and thus was selected for use with some normalizing factors.

The power vector described above is normalized from a range of 0 to 1 based on the maximum power value in the vector, as shown in Figure 3.6. After this, the geometry array is modified to have solid material around the edges, and creates a 51 × 11 grid. After this is completed, the normalized power is used to generate the first layer of material. The code goes

through the first row bit by bit, and reads the corresponding normalized power value. A probability is applied to the bit location to determine if a solid bit is generated. This probability ranged from 15-80% based on the normalized value of the power. In other words, a location with a normalized power value of 0 has a 15% chance of generating a solid bit, while a location with a normalized value of 1 has an 80% chance of generation. Any value in between uses a linearized relationship between the percentage and normalized power value. Percentage values utilized were set through trial and error testing, the values chosen are intended to bias the solid bits toward the regions of highest power. The percentages chosen gave the most effective result in solid bit generation.

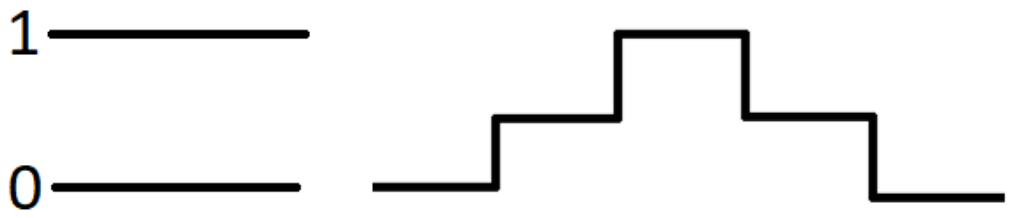

Figure 3.6: Example of a normalized power from the first analysis.

Once the first layer is generated, the rest of the geometry can be created. Unlike the first layer, the remaining layers chance of generation is not reliant on the power map, but based on surrounding material and a 50/50 probability of generation. Power is not directly utilized as it assumed the first layer provides the necessary bias in the structure. At the start of this flow, the code goes to the new layer directly above the first layer, and runs from the left edge to the right in the row. At each point, it is determined if there is any solid material around the point (including the walls on the side). If there is any material in one of the 8 surrounding cells, the current location is given a 50% chance of generating a solid bit. After this, the next bit location experiences the same operation, and this repeats until the vector is completed. The process repeats with the next layer, starting back on the left edge. After one pass-through in each row is

completed, the geometry is finalized. An example of this geometry is shown in Figure 3.7. While creating the generation code, a test was done to compare the generated geometry with one generation pass through versus many. The single pass through was chosen in the end, as the multiple pass through (even with reduced generation chance) created large solid structures in the center of high power zones, essentially creating flow obstructions.

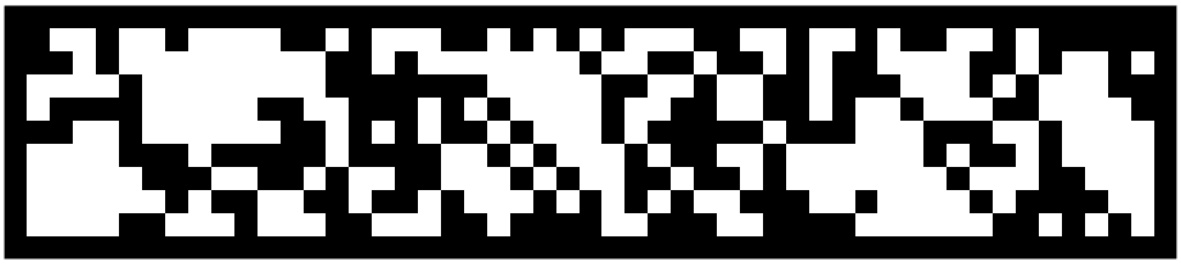

Figure 3.7: Generated design bit array

3.3. Solid Model Creation

Once the bit array representation of the geometry has been created, it needs to be converted into a form that can be read by the ANSYS software package, DesignModeler (2016a). The package requires a formatted text file, containing positional data of points, which are used to create closed curve structures. To create this file in MATLAB, it was determined the best way to accomplish this was to start with a contour plot function of the array. The contour plot is easily created with a single function (and the information stored in variables), but it is important to note that two contour levels need to be used for the plot. Two contour levels are needed to ensure generated curves have realistic geometry. The first initial contour level often leaves bits floating in space, or completely misses material connections. This is overcome by utilizing the second level, in exchange for some dimensional accuracy (approximately a 0.125 mm increase in material thickness, max). An example of the contour and corresponding bit array are presented in Figure 3.8. With the contour plot and information created, the program utilizes the script C2xyz,

available on the Mathworks site by author Chad Greene (2014). This script organizes the information from the contour variable into x and y coordinates of the readily extractable groupings. The program then takes this information, and separates the curve groups and coordinates into a text file that can be read by ANSYS software. Before finishing operations, the program prints out the curvature file discussed above, a picture of the bit array, a picture of the contour plot and a CSV containing the bit array information of the geometry.

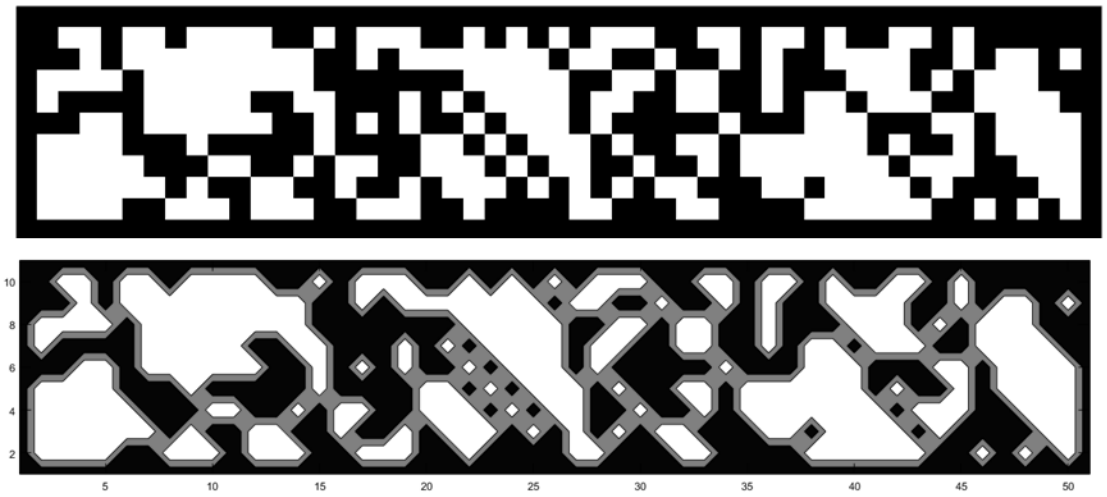

Figure 3.8: The generated bit image and corresponding contour plot. Note that the black in the contour plot represents the results of the first contour line, while the grey represents the second contour line.

3.4. ANSYS Package and Computational Fluid Dynamics

With a curve file available and design information saved, work to solve the problem in the CFD solver can begin. ANSYS Workbench (2016) is the main program that is used, with the subprograms of DesignModeler (2016a), Meshing (2016) and Fluent (2016) handling the operations. The operational flow laid out in Workbench goes from Geometry to Mesh and lastly to Fluent (see Figure 3.9). DesignModeler is used rather than SpaceClaim (also part of the

ANSYS package), purely because of its curve functionality which allows custom shapes to be imported quickly.

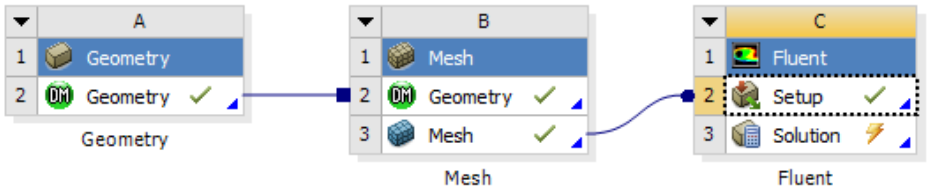

Figure 3.9: Workbench block flow

Starting with DesignModeler in Geometry, the screen that greets the operator is shown in Figure 3.10. Now the system geometry is created. This is the 52 × 12 mm external area of the heat sink device, though the corner is offset from the origin by 1 mm in each direction. This is to ensure that the internal flow area is oriented at 0, 0 so that the grown geometry can be accurately placed. After this is created, the geometry is extruded 50 mm in the -Z direction. Then, the flow channels are created. This is started by placing a 1 mm thick wall around the edge of the part (leaving the internal area completely open), and extruding this wall by 5 mm on each end. This wall, and the chosen extrusion size was to give the flow some space to develop around the entrance, allow for fluid connections in a real device, and to minimize effects as a result of mass added to the device. Now the generated geometry from before is imported. This is done by creating a curve operator on the interior surface of the heat sink, then designating the operator to get information from a coordinate file. The coordinate file generated in the previous steps is selected, and the geometry generated. An extrusion operation is created, and the curves from the previous operation selected. The extrusion command is then used to cut the geometry out of the stock material. A fill operation is performed by selecting all of the internal surfaces of the device. The fill serves to create the geometry of the fluid path, and is necessary to perform any CFD operations. Finally, a plane is located at the bottom of the device. A small 50 mm × 50 mm

surface is created on this plane, directly below the internal area of the device (i.e. 1 mm from the sides and 5mm from the ends). This surface is extruded 0.1 mm from the geometry, and serves as a physical location for the heat transfer map to be applied. The device is now created, an example of which is seen in Figure 3.11. DesignModeler is now closed.

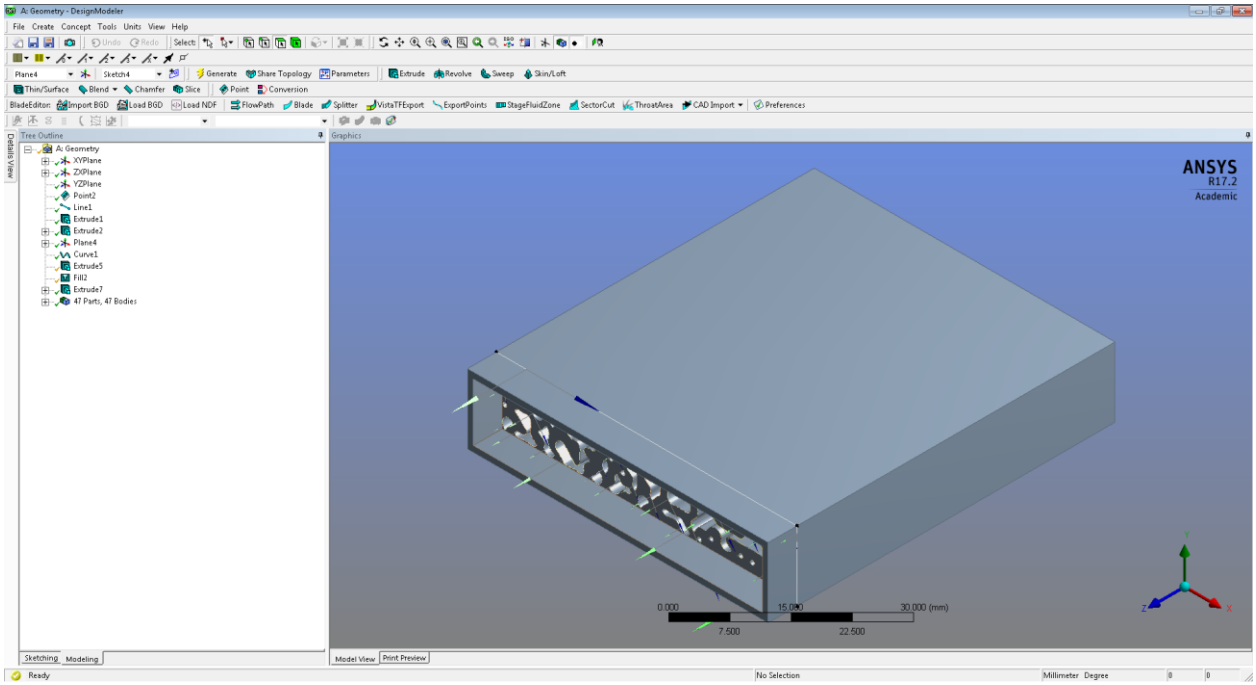

Figure 3.10: View of the DesignModeler software, with a generated solution

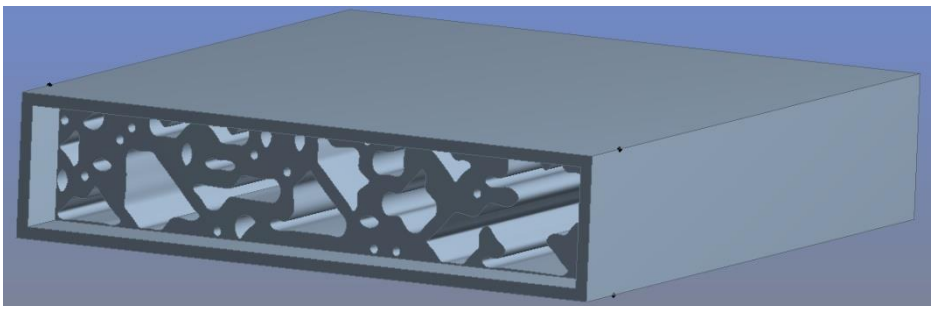

Figure 3.11: Final device created in DesignModeler

With the geometry created, the system needs to be meshed before the CFD solver can be used. The built in ANSYS Meshing software is used rather than the Fluent meshing software for simplicity and speed in updating the mesh. Meshing is opened, presenting the screen shown in Figure 3.12, and the mesh is defined. A separate sizing is created for the solid, allowing control

over both the fluid and solid mesh densities. Mesh settings are mostly left at the default values, with the exception of the relevance center and smoothing set at medium, and the minimum grid size for both solid and fluid geometries set at 0.5 mm, a representative mesh is shown Figure 3.13.

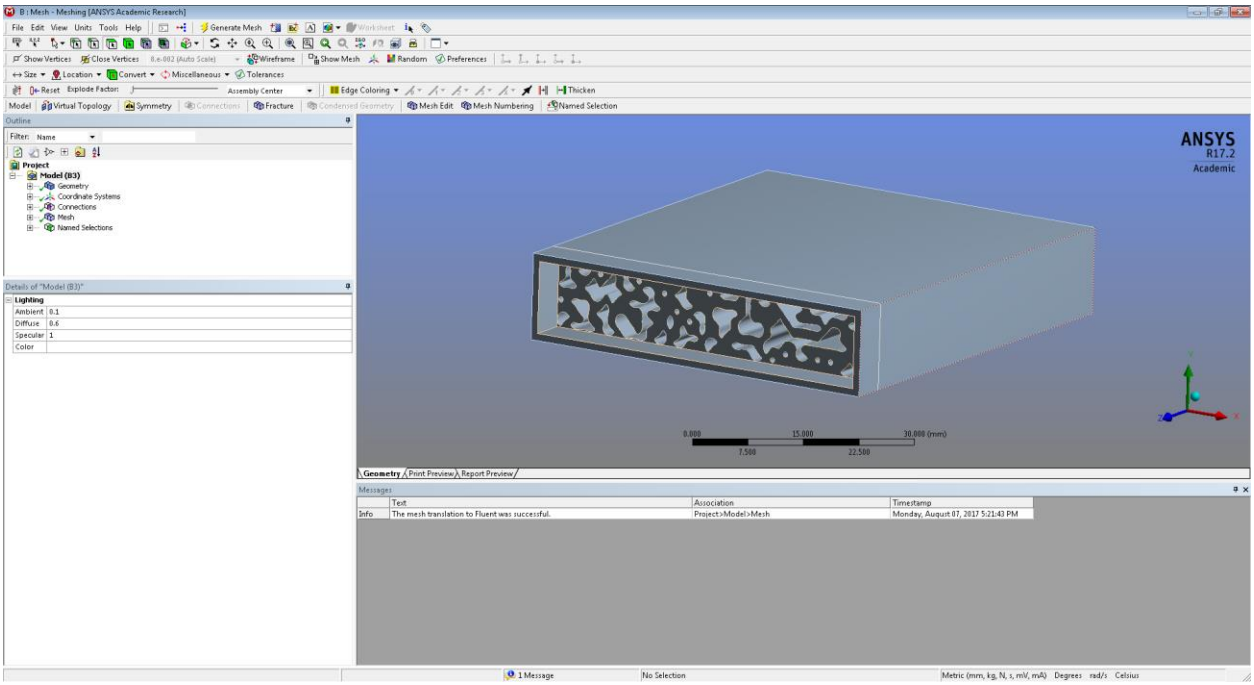

Figure 3.12: View of the meshing software

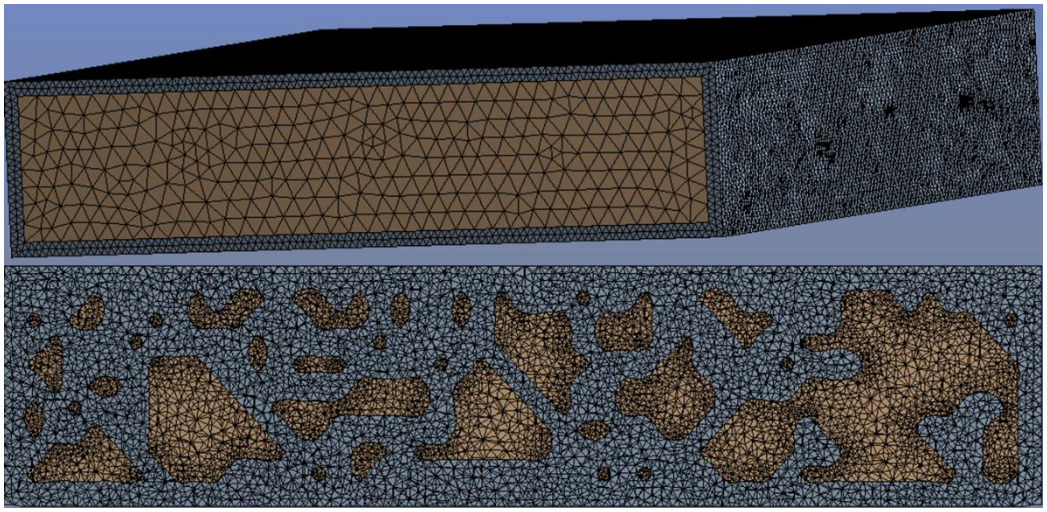

Figure 3.13: View of the meshed device, as well as the meshed cross-sectional view.

The mesh size was selected after performing a Grid Convergence Index (GCI) study to determine the best balance between the fineness of the grid and analysis speed. Tests were done with both the fluid and solid geometry, with three different grid sizes for each. For the solid, tests were done with a minimum grid size of 0.25, 0.50 and 1.0 mm and analyzed the difference in system temperature to determine convergence. The fluid is analyzed with slightly smaller grid sizes at 0.125, 0.25 and 0.50 mm and total pressure drop is used as a convergence criteria. Analysis is provided in the table below, and showed that the largest GCI difference was within 1.5%. This means that the difference in the grid sizing will not have a significant impact on the system result. In addition, the finest grid refinement takes more than 24 hours to run a single simulation, which is not practical for running the large number of analyses required for genetic algorithm optimization. However, the largest grid refinements can sometimes result in a failure for the mesh to generate, as a result of the grid being larger than feature sizes. It was decided that a minimum size of 0.5 mm for both the solid and fluid would be used, as it provides good accuracy and reasonable analysis times, while also providing a low instance of meshing failure.

Table 3.1: The step size and results for both the solid and fluid grid refinements.

Step Size (Solid)	0.25	0.50	1.0
dT (K)	18.39	18.23	19.28
Step Size (Fluid)	0.125	0.25	0.50
dP (Pa)	121.8	121.4	119.9

Table 3.2: The results of the GCI analysis

GCI (Solid) 0.25/0.5 (%)	0.2067
GCI (Solid) 0.5/1.0 (%)	1.335
GCI (Fluid) 0.125/0.25 (%)	0.1340
GCI (Fluid) 0.25/0.5 (%)	0.5266

Back in Workbench, CFD analysis can now be started from the Fluent -Setup tab. The Fluent screen is shown in Figure 3.14. The power profile is imported by going through “File”, “Read”, “Profile” and navigating to find the text file generated at the beginning of this section. Next, the viscous model was set from laminar to Spalart-Allmaras (1-Eqn), and the energy model enabled.

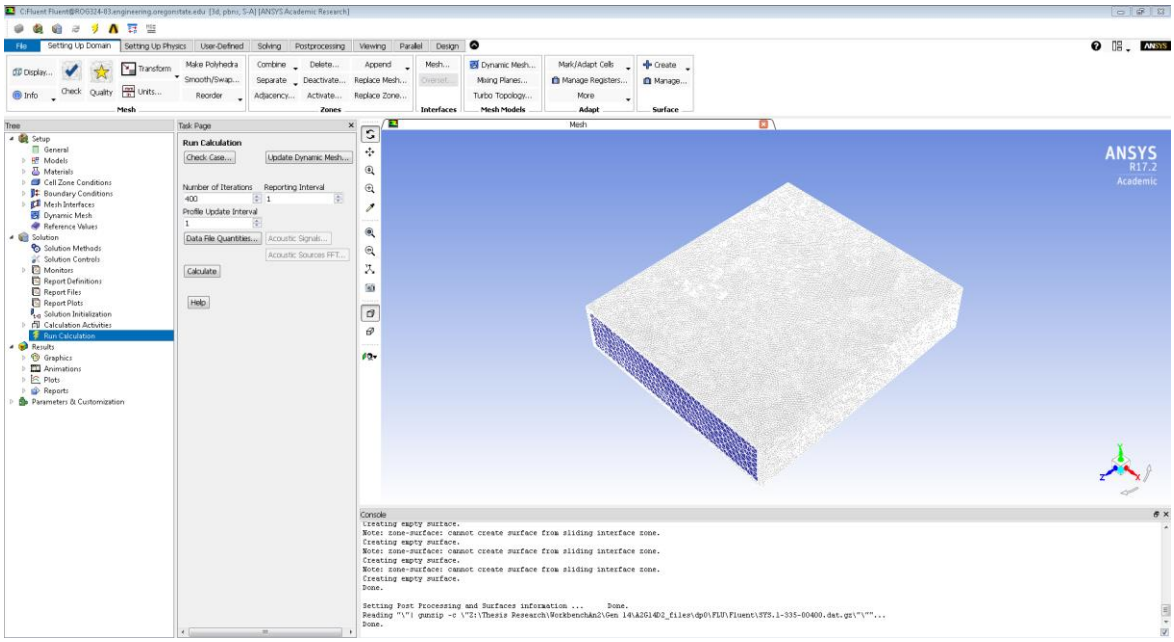

Figure 3.14: View of Fluent utilized for CFD analysis

The Spalart-Allmaras transport equation is shown below (3.1).

$$\frac{\delta}{\delta t}(\rho\bar{v}) + \frac{\delta}{\delta x_i}(\rho\bar{v}u_i) = G_v + \frac{1}{\sigma_v}[\frac{\delta}{\delta x_j}\{(\mu + \rho\bar{v})\frac{\delta\bar{v}}{\delta x_j}\} + C_{b2}\rho(\frac{\delta\bar{v}}{\delta x_j})^2] - Y_v + S_v \quad (3.1)$$

Here the transported variable, $\bar{v}$, is identical to the turbulent kinematic viscosity except in the near-wall region. G_v is the production of turbulent viscosity, while Y_v is the destruction of turbulent viscosity near the wall. σ_v and C_{b2} are constants, while $S_{\bar{v}}$ is a user defined constant. μ is the dynamic viscosity of the fluid and ρ is the density (ANSYS, 2006).

Next the energy model is provided below (3.2). Energy (*E*) and Temperature (*T*) are treated as mass-averaged variables, in (3.3).

$$\frac{\delta}{\delta t}(\rho E) + \nabla\bullet(\bar{v}(\rho E + p)) = \nabla\bullet(k_{eff}\nabla T) + S_h \quad (3.2)$$

$$E = \frac{\sum_{q=1}^{n}\alpha_q\rho_q E_q}{\sum_{q=1}^{n}\alpha_q\rho_q} \quad (3.3)$$

Again, ρ is the fluid density, while p is the local pressure. The effective thermal conductivity is represented by k_{eff}. S_h is as a source term, with contributions from radiation and other volumetric heat sources. In the mass-averaged energy equation, each variable is based on the specific heat of the phase and a shared temperature. The thermal diffusivity is represented by α_q, while ρ_q again represents the density. (ANSYS, 2006)

The Spalart-Allmaras model is used rather than the laminar model to account for the potential for turbulent flows in some portions of the heat sink, while still minimizing solving time over other turbulent models. Unfortunately, this will make the laminar sections less accurate

in order to account for potential turbulent sections, as most turbulent models aren't tuned to laminar solutions. The results between the laminar and turbulent models typically see about a 3-4% difference in predicted pressure drop (see Table 3.3). It was determined that for the purposes of this optimization investigation, the Spalart-Allmaras model handles laminar problems reasonably, while also allowing calculation of turbulent regimes. The default constants and Prandtl values remain for the Spalart-Allmaras model.

Table 3.3: Comparison of the pressure drop result in a laminar vs Spalart-Allmaras model in a few solutions.

dP (laminar) (Pa)	dP (turbulent) (Pa)	% Difference
73.71	76.39	3.635
55.06	56.49	2.601
127.1	133.9	5.316
43.41	44.13	1.659
108.6	113.2	4.453
58.77	60.28	2.571
79.08	81.59	3.168
129.4	135.5	4.694
75.91	78.09	2.878
	Average	3.442

Now materials for the solid and fluid are chosen, which are aluminum and water (liquid) respectively. Boundary conditions are set, with the inlet being a set as a velocity inlet (with an absolute velocity set uniformly across the entrance area, the profile is affected in the model), the outlet as a pressure outlet and the bottom surface as a power boundary. Leave the outlet settings as defaults, but adjust the inlet velocity to 0.1 m/s. Adjust the bottom surface to make use of the power profile imported earlier. Now, a mapped interface needs to be created to make the energy model work. A mapped solution method is created using the default SIMPLE method. The solver is set to run 400 iterations, which achieves convergence in the system.

The typical solver results are shown in Figure 3.15, below. After 400 iterations, the solution residuals of velocity, energy and the turbulent kinetic viscosity are on the order of 10^{-6} and 10^{-5}. Continuity, however, is larger, on the order of 10^{-3}. The ANSYS Fluent user guide (ANSYS, 2006) suggests having results on the order of 10^{-6} is a standard target for residuals, meaning the former are in the expected range. It also suggests that in some pressure based solver, residuals may only approach 10^{-3}, which seems to be what is occurring with the continuity equations. Based on the ANSYS documentation, these residuals were determined to be reasonable.

After the solver has ran, the results are collected into a spreadsheet, an example is provided in Figure 3.16. The data collected from the system starts with the power input into it. The actual value of power is 700.1 W in the first analysis (symmetric map), and 218.7 W in the second (non-symmetric map). The energy balance of the system should be determined for each solution, as a measure of how accurate the solution should be. The result should be within 1% of the actual value, otherwise the solution should be reinitialized. Next, the volume integral temperature of the fluid is recorded, which provides the average volume temperature of the fluid. This is effectively the mixed cup temperature of the fluid. Pressure drop is recorded between the inlet and outlet, and at the bottom surface, the maximum temperature.

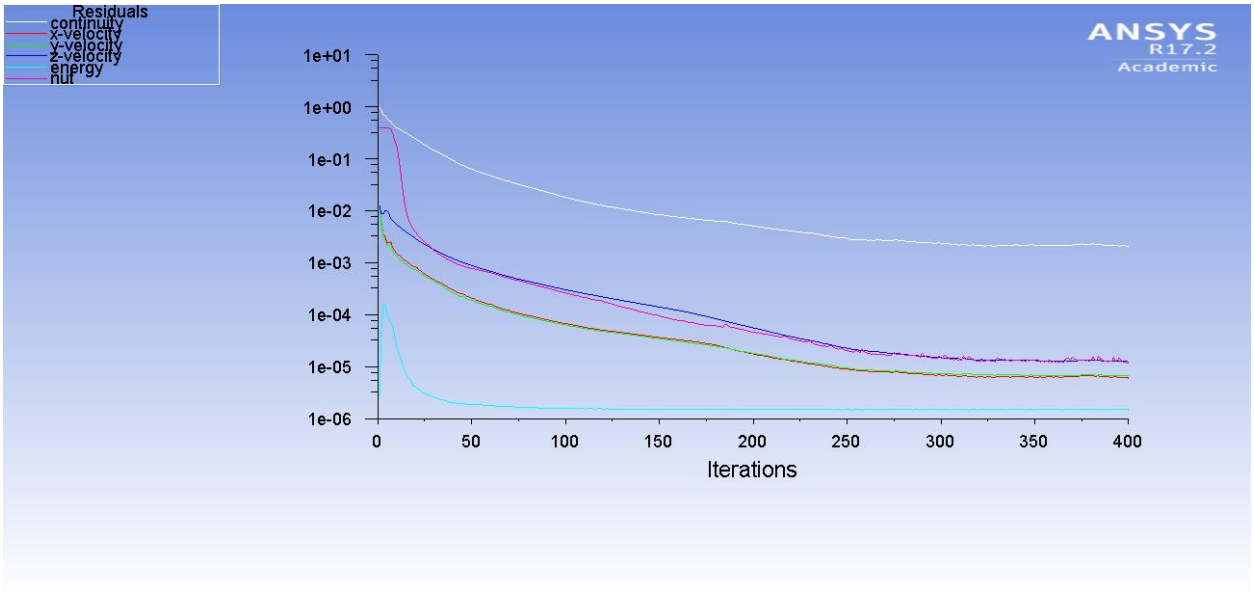

Figure 3.15: Solution residual values after 400 iterations

	A	B	C	D	E	F	G	H	I	J	K
1		dP	Surface Ma	Water Avg	Energy Balance		dT_max		R_avg	dP	S_gen
2	5.11	391.6851	309.2389	300.6561	218.6424		8.5828		0.039253	391.685145	0.020185
3	2.12	141.5743	312.7251	300.7624	218.6667		11.9627		0.054711	141.574337	0.02781
4	3.12	234.2859	309.8098	300.7155	218.682		9.0943		0.041592	234.285882	0.021344
5	4.12	261.1222	311.5097	300.7557	219.522		10.754		0.049183	261.1222225	0.025098
6	5.12	242.2512	311.225	300.7233	218.7244		10.5017		0.048029	242.251241	0.024534
7											
8	Q''	Q	m_dot	c_p	rho						
9	87460.97	218.6524	0.04991	4183	996.6						

Figure 3.16: Example of the table used to record results

With the results collected, they need to be organized and used to determine the entropy generation rate of the solution, and the thermal resistance based on maximum heat sink temperature. This was also discussed in Chapter 2 (2.15).

$$R_{th} = \frac{T_b - T_\infty}{\dot{Q}_b} \tag{2.15}$$

Also discussed in Chapter 2, an expression for entropy generation rate (reproduced in Eq. 3.4, below) was derived by Bejan (1982), and is used in the analyze the data present study.

$$\dot{S}_{gen} = \left(\frac{\dot{Q}_b^2}{T_\infty T_b}\right) R_{th} + \frac{\dot{m}\Delta P}{\rho T_\infty} \tag{3.4}$$

The baseline values utilized in the calculations are shown in Table 3.4, for both analyses.

Table 3.4: System values in calculations

Property	**Analysis 1**	**Analysis 2**	**Units**
Power Input	700.1	218.7	W
Average Heat Flux	28.00	8.746	W/cm^2
Ambient Pressure	101300	101300	Pa
Ambient Temperature	300.0	300.0	kg/m^3
Fluid Density	996.6	996.6	kg/m^3
Mass Flow Rate	0.04991	0.04991	kg/s
Heat Capacity Ratio	4183	4183	--

With the current generation analyzed, the results need to be updated in a few files before the new generation can be created. A CSV file needs to be updated with information from the analysis, and is displayed in Figure 3.17. Additionally, a generational results file is updated, by replacing the file and array information in the generational data folder with the new data. Similarly, the memory needs to be checked to determine if a solution was replaced, and the appropriate array information and memory file updated. Once completed, MATLAB can be opened to create a new generation.

	A	B	C	D
1	5.11	0.039253	391.6851	0.020185
2	2.12	0.054711	141.5743	0.02781
3	3.12	0.041592	234.2859	0.021344
4	4.12	0.049183	261.1222	0.025098
5	5.12	0.048029	242.2512	0.024534

Figure 3.17: CSV file that is imported into MATLAB.

3.5. Optimization Code

With the results from the previous generation compiled, and the folders populated with the correct information, a new generation can be created. In MATLAB, the active folder is directed to where all the files and CSV data are stored. The script first starts by setting the generational number (example: the next generation should be #18) and the memory size. It is important to update the generational number with each successive generation, otherwise this will create problems with the external memory. The memory size should only have to be set once. In this test, the first analysis memory array was set to 25 samples, meaning that the first generation will require 25 designs and analysis. The second analysis was set at 20 samples, to determine if there were any benefits to changing the size. With those set, the script starts looking for the generation and memory files and imports them with the corresponding array data. The system also collects performance information from the current generation memory, as well as the external memory from the "GenData.csv" and "ExtMem.csv" files respectively. Once the information is initialized into the system, the systems modifies the collected array information by adding surrounding walls, which are necessary for geometry modification.

Now the algorithm work can start. The system checks the generational data (or memory if this is the first run), and ranks each solution from best to worst based on entropy generation. The generation solutions can now be compared to the memory (skipped on the first run). Two numbers between 1 and 25 are randomly generated to serve as access numbers to the memory. A check loop ensures that the numbers are not the same. The associated memory locations are compared to the #2 and #4 ranked solutions via the entropy generation rate. As discussed previously, if the generational solution has a lower entropy generation rate than the memory solution, it will be recorded as replacing it in the final memory output. If the memory solution

has the lower value, it remains in the memory and the solution array replaces the solution in the current generation. Once the final generation state has been created, the new generation can be created. This process is illustrated in Figure 3.18.

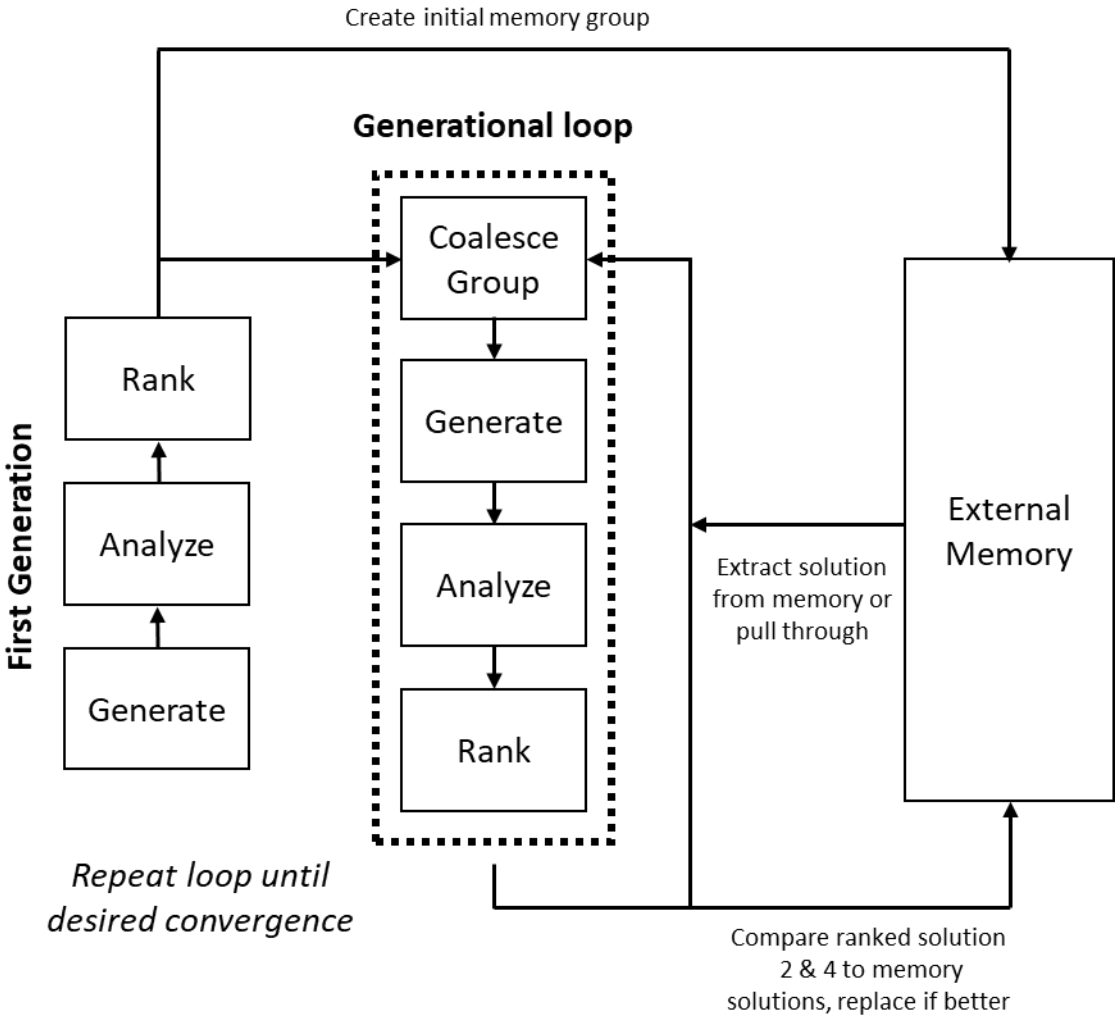

Figure 3.18: Diagram of the genetic algorithm flow

The first ranked solution is treated as the elite individual, and thus is not modified in the new generation. For the rest of the solutions, a crossover occurs first, and creates crossovers between 1-2, 2-3, 3-4 and 4-5. When the operator initializes, it copies both parent arrays to its own memory, and initializes the new child using the first parent as a base. Similar to the

generation technique, the script runs through row by row until it reaches the bottom. At each point, a random number is generated and this determines if the second parent provides the data for that point. There is a 50% chance that the second parent will be used. If a solid pixel will be added to the space, it follows the same rules as point generation. The pixel must have another solid pixel around it, or it will not generate. Deleting a solid pixel is a much more difficult issue, as it is very easy to create geometry that cannot physically exist. For example, if three pixels extend from the edge of a solid part, and you delete the first pixel that starts the extension, the two end pixels will just remain in space. To get around this issue, a few filters are used in the system. In this instance, rules are set to determine what can be deleted, similar to the requirements on what can be added. If there are two solid pixels around the current pixel, then the system is not allowed to delete the current position, as shown in Figure 3.19. This is done to prevent the system from deleting bridges between shapes. While there is some loss of accuracy, it is a necessary tradeoff to produce realistic designs.

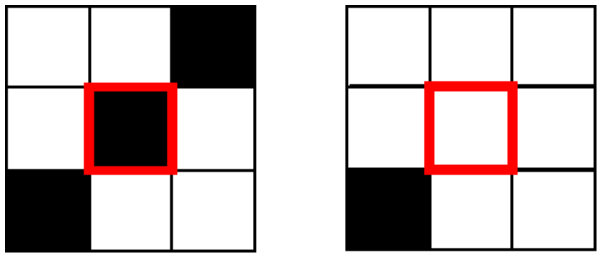

Figure 3.19: On the left, an example of a solid bit which cannot be deleted, with two surrounding bits. The right, a bit location that a solid bit may be added to.

With the crossover operation completed, mutation can be performed. The system goes through each pixel in the array, and will check the mutation percentage against a randomly generated value. If the value is less than the percentage chance, the pixel bit in the current position will be flipped. That is, if it is a solid pixel, it will be changed to empty space, and if it's empty space, it will become a solid pixel. At the same time, the pixel change still must follow the

rules laid out previously for both creating a solid pixel, and deleting a pixel. With mutation done, all the algorithms are complete. The final population solution operators are shown in Table 3.4. A filter is now utilized to hopefully clear any remaining outliers in the array. The filter is very simple in design and execution; it goes through each locational point and looks at the surrounding pixel count of every solid pixel. If there are 2 or less solid pixels surrounding, the selected solid pixel is deleted. A balance had to be chosen for this filter, as just deleting shapes with only one surrounding pixel will leave a lot of the extraneous shapes in the final array. Any more, and a large amount of valuable information will be lost. In the end, two was selected as it offered a good balance of deleting bad bits, while not sacrificing a large amount of functional data.

Table 3.4: Table showing the operations performed on each solution

Solution #	Operators
1	Elitism
2	Crossover 1-2, Mutation, Filtered
3	Crossover 2-3, Mutation, Filtered
4	Crossover 3-4, Mutation, Filtered
5	Crossover 4-5, Mutation, Filtered

With the final arrays created, the new generation information can be exported. The script follows the same techniques in previous codes, but uses loops to export all the information at once. It creates a binary image of the array, an image of the contour plot, exports the contour curve file, creates a base CSV file with the generation and solution number to copy results into, a copy of the randomized numbers used for memory comparison and, finally, copies the final memory array into a CSV file. With all of this printed out, the information can be copied into the appropriate generation file. The process repeats with importing the information into ANSYS.

The solid model and subsequently the mesh is updated for the new solutions, and another Fluent analysis is run.

3.6. Genetic Algorithm Settings

The algorithm used in the problem utilizes a combination of elitism, mutation and crossover. As discussed by Deb and Agrawal (1999), analysis convergence occurs fastest when utilizing a combination of all 3 functions. Population size is similar in size to other micro-genetic algorithms (Coello & Pulido, 2001), with 5 total solutions and 1 providing solution elitism. The other 4 solutions are acted upon by both mutation and crossover operators. Crossover operates fairly simply, with each parent having a 50% chance to contribute to the child, at least if contributing a solid bit. Because of the reduction of occurrence from filtering, the chance for a solid bit to be deleted is set higher at 65%. This is done to try and provide a balance to the crossover that is disturbed as a result of the filtering methods. On a similar note, the mutation rate uses different mutation percentages based on if a bit will be added or deleted. If a solid bit will be added, the percentage chance for mutation is set at 30%. If a solid bit will be deleted, the percent chance of mutation is set at 40%. Again, the deletion chance is set higher to balance the limitations set by the filtering system. Of note, the mutation values were set similarly to those utilized by Tai *et al*. (2005), who found mutation rates of 30-40% produced the lowest relative error with bit-array problems. Unfortunately, it is difficult to ascertain the ideal values for genetic operators, which typically requires a parametric study to determine (Deb & Deb, 2014), which was beyond the scope of this study.

4. Results and Discussion

The performance of every generated profile is compared against a standard finned heat sink. The straight finned geometry is placed into the same space as the generated designs (see Figure 4.1, below). The finned heat sink has 19 interior fins, 1 mm thick, with a spacing of 1.55 mm between each fin. Again, this is based on the design from Remsburg (2007) and their water impingement design. The pressure drop of the system at a flow rate of 0.1 m/s is calculated as 59.34 Pa. Utilizing the first (symmetric) power map, it is found that the entropy generation rate is 0.2649 W/K. The second, non-symmetric, power map results in the generation rate being 0.02445 W/K, as less heat is transferred. The mass of the device was determined to be 46.76 grams assuming a material density of 2720 kg/m^3. These values serve as the baseline that all generated designs are compared to.

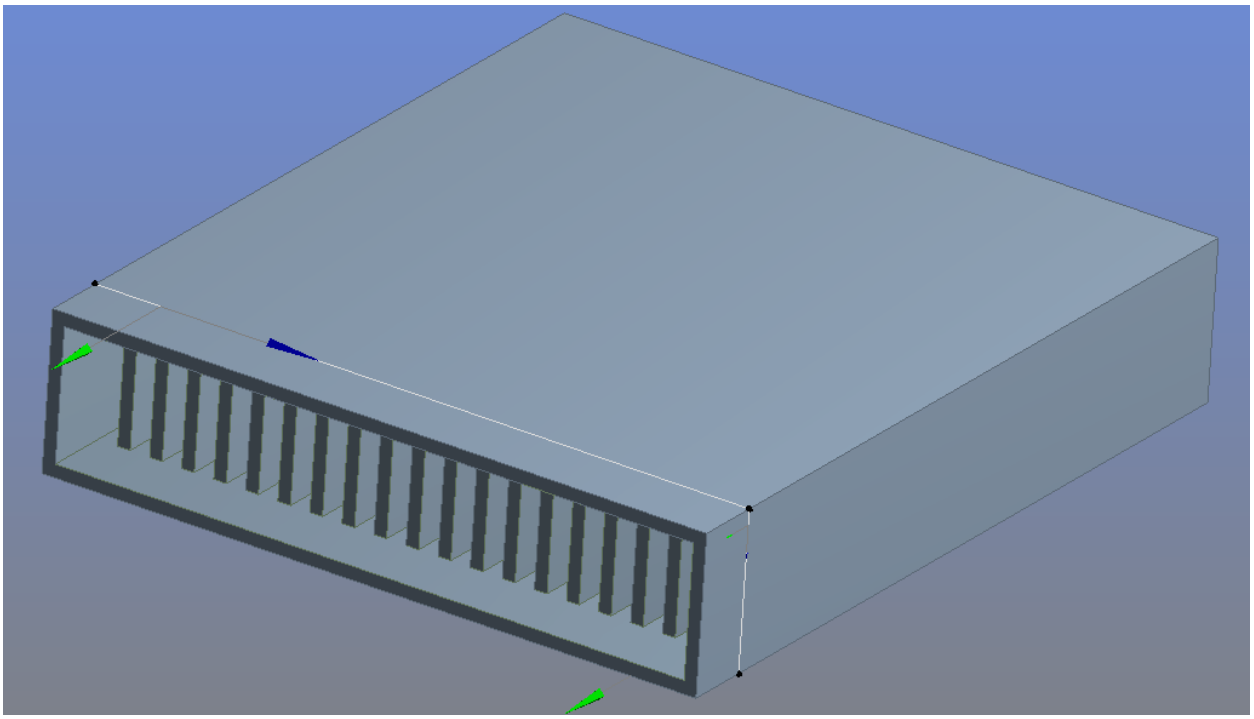

Figure 4.1: Image of the finned heat sink used for comparison purposes

4.1. Symmetric Power Map Results

Before the results are discussed, it should be noted that an error was encountered during the initial generation attempt. This error was from the algorithm not correctly implementing the elitist solution. As a result, the elite solutions were all based off of the first best result. It was discovered after 38 generations were completed, so it was decided to leverage the results to regenerate and solve the corrected solution set. This was done to determine the effects of a strong initial population on the algorithm process. To better determine how quickly the ideal solution may emerge from a typical scenario (i.e., with no strong initial population), the analysis with the non-symmetric power (discussed in Section 4.2) was started with no initial seeding.

The symmetric power map was reinitialized after 38 generations. Each of these generations contained 4 unique solutions (from crossover and mutation operators), and 1 elite solution carried over from previous generations. The exception was the first generation, which contained 20 unique solutions. To reinitialize the problem, 25 of the best solutions from the first solution set were carried over as the first generation of the new generation set. From here, generations remained as 4 unique solutions and 1 elite solution. One hundred generations were ran before analysis was stopped. This number was chosen based off the result of the best solution, which saw little change after approximately 75 iterations. Going to 100 helps ensure that the solution values have were optimal. With this done, the data is analyzed.

The five lowest entropy generation rates were determined to be 0.1992, 0.2033, 0.2078, 0.2082 and 0.2085 W/K. Compared to the finned heat sink at 0.2649 W/K, this is a reduction in entropy generation rate between 21 and 25%. Similarly, these five designs had a reduction in thermal resistance from 24 to 27% compared to the baseline. This also corresponds to a drop in maximum surface temperature of up to 11.27 K. The reduction in thermal resistance is

accompanied by an increase in device mass (the finned device weighing 46.76 g), as well as an increased pressure drop compared to the fin design (59.43 Pa). The solution masses were 69.37, 65.38, 64.79, 66.78 and 64.63 grams. This means the mass increases anywhere from 38.22 to 48.37%. It is important to note that the mass was not an optimization target in this research, so it is expected that it would be impacted. However, it is still of interest in regards to other variables such as production cost, and remains a potential goal in future optimization efforts. In regards to the pressure drop, the top five generated designs have a pressure drop of 349.0, 261.4, 226.7, 290.4 and 216.0 Pa respectively. This is a respective increase of 487, 340, 281, 389 and 264%. This is a noticeable increase over the finned system, and is best explained by the relative irreversibility due to pressure loss of the working fluid compared to heat transfer over a finite temperature difference. Because the pressure drop is on the scale of Pa, a significant change on the order of 100 Pa will not have the same effect as a change of a few Kelvin on the maximum temperature for a fixed heat input. Thus, from a thermodynamic perspective, the overall heat removal process is more reversible in the generated heat sinks, despite the increased pressure drop. However, practically, the thermodynamic improvement would need to be balanced with potentially higher pump operating costs.

The highest power vector utilized for the solution is displayed in Figure 4.2, along with the contour figures of the best five solutions. From initial inspection, the largest mass of material is in the center of the cross section, with some extension toward the edge. Mass tends to start at the base center, and then branch out further, conducting heat away from the hottest portion of the chip. The far edge of the device sees less overall mass than the center areas. Most of the open area occurs around the far edge and the top of the device. To validate these initial observations, the solutions were subdivided into 5 sections, with the fill in the areas calculated and shown in

Figure 4.3. Sections 2, 3 and 4 represent the center of the device, and it can be noted that the fill densities are consistently within 50-55% for these areas. This is in contrast to the outer edges analyzed by sections 1 and 5, where the fill density is typically closer to 45-50%. This confirms that the system is biasing solid material toward the center of the devices. Total fill density of the devices was also determined, at 53.97, 50.11, 49.21, 52.15 and 49.21% for the respective solutions. This will provide an interesting point of comparison to the fill density of the non-symmetric solutions.

The temperature maps shown in Figure 4.3 help present some of the effects of the solutions. Results are as expected; the high-power area of the map causes a semi-elliptical temperature distribution across the surface base around the center. The additional material around this area helps to dissipate the energy, decreasing the maximum temperature of the device. An obvious problem results from the diagonal openings in the solution, as represented by the very small holes throughout the solution space. Notable in the temperature distributions, these areas maintain the temperatures of the surrounding material. This indicates the openings serve little purpose in terms of the flow, and would be better replaced by material. The added mass from the material would help further dissipate the energy, and because of the limited area for flow, would have negligible effect on the pressure drop of the system.

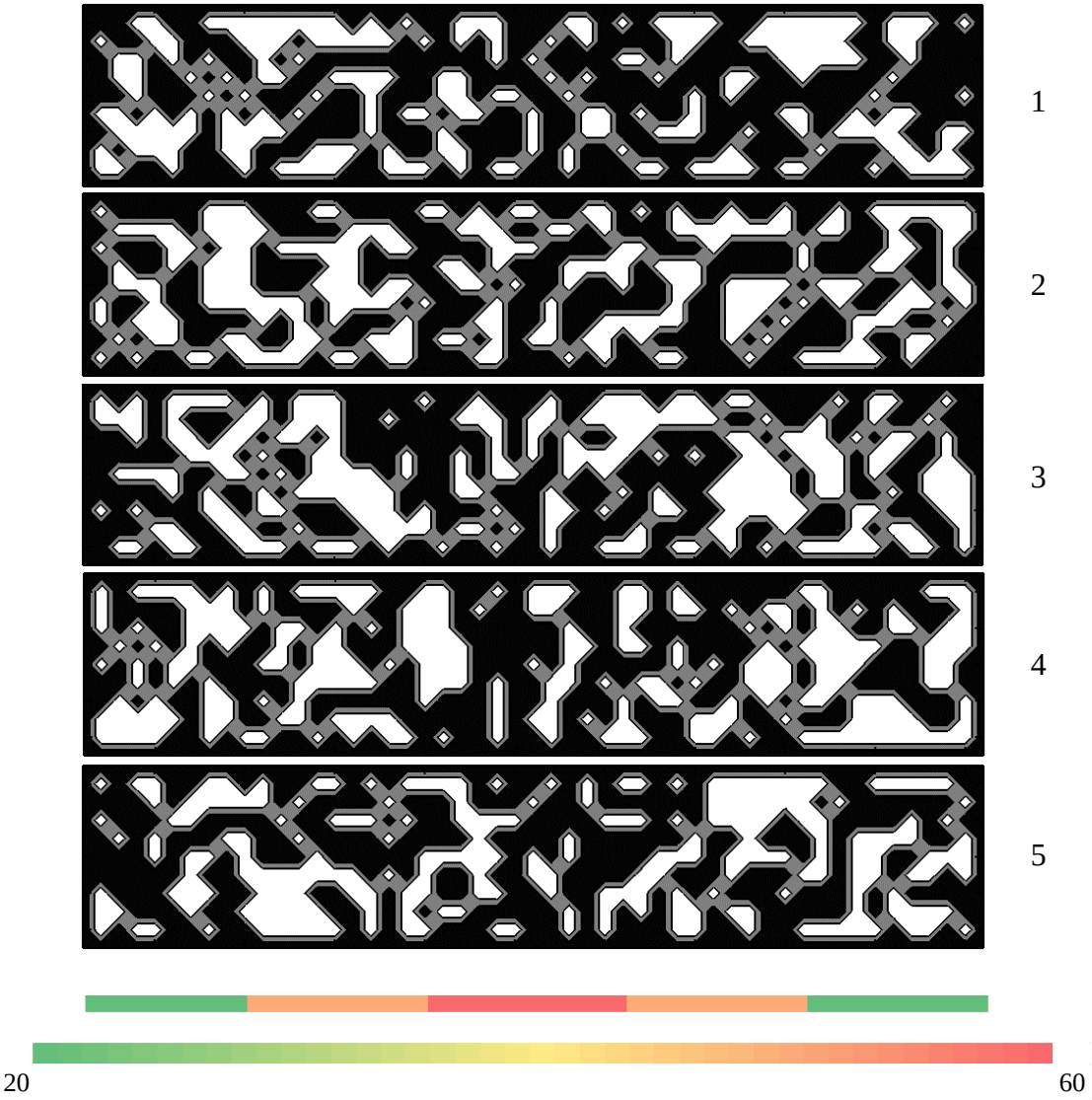

Figure 4.2: Contour figures of the top five solutions providing the lowest entropy generation rate utilizing the symmetric power profile. The power profile is provided below, measured in W

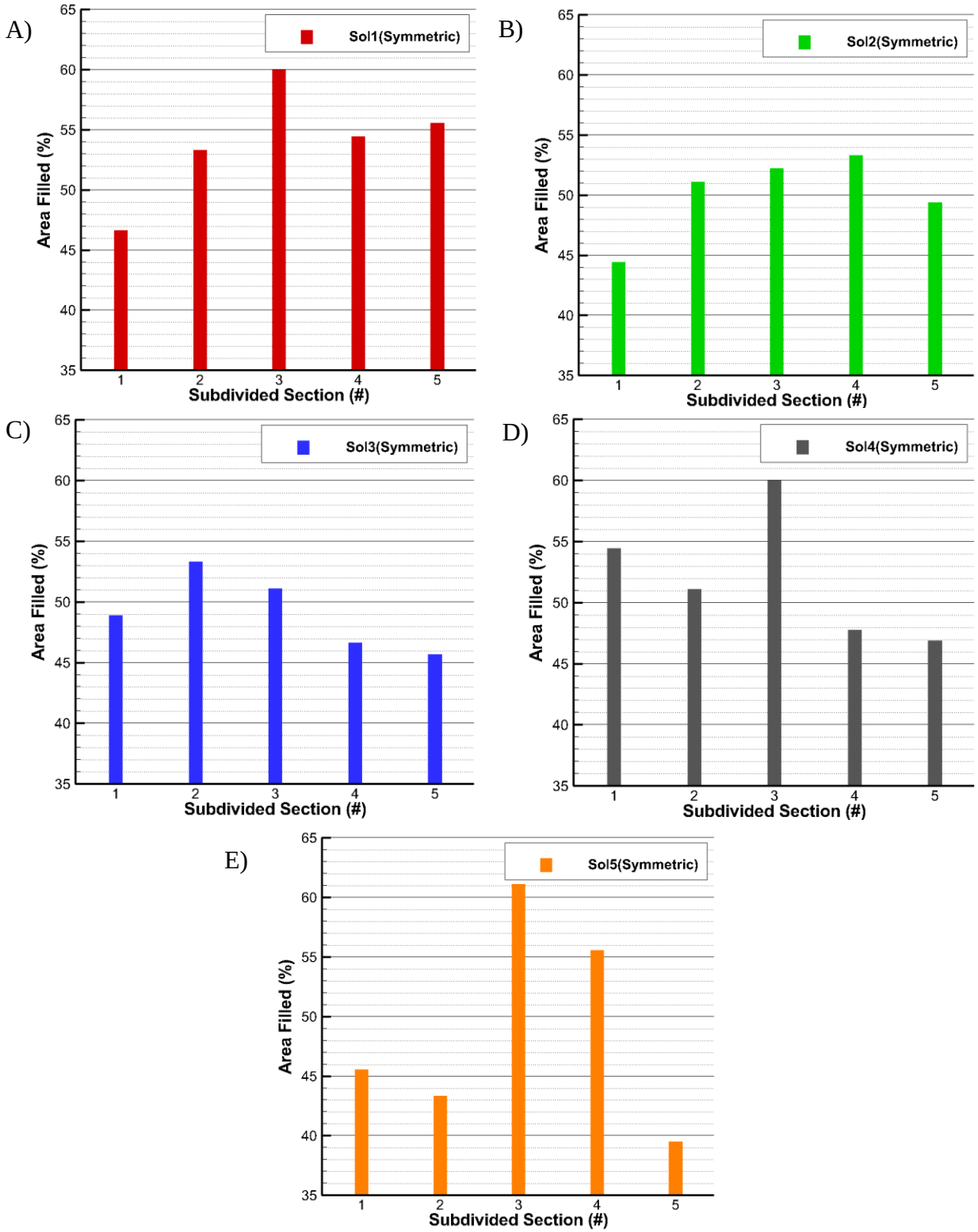

Figure 4.3: Fill densities of the subsections of the symmetric solutions. A) Solution 1, B) Solution 2, C) Solution 3, D) Solution 4, E) Solution 5

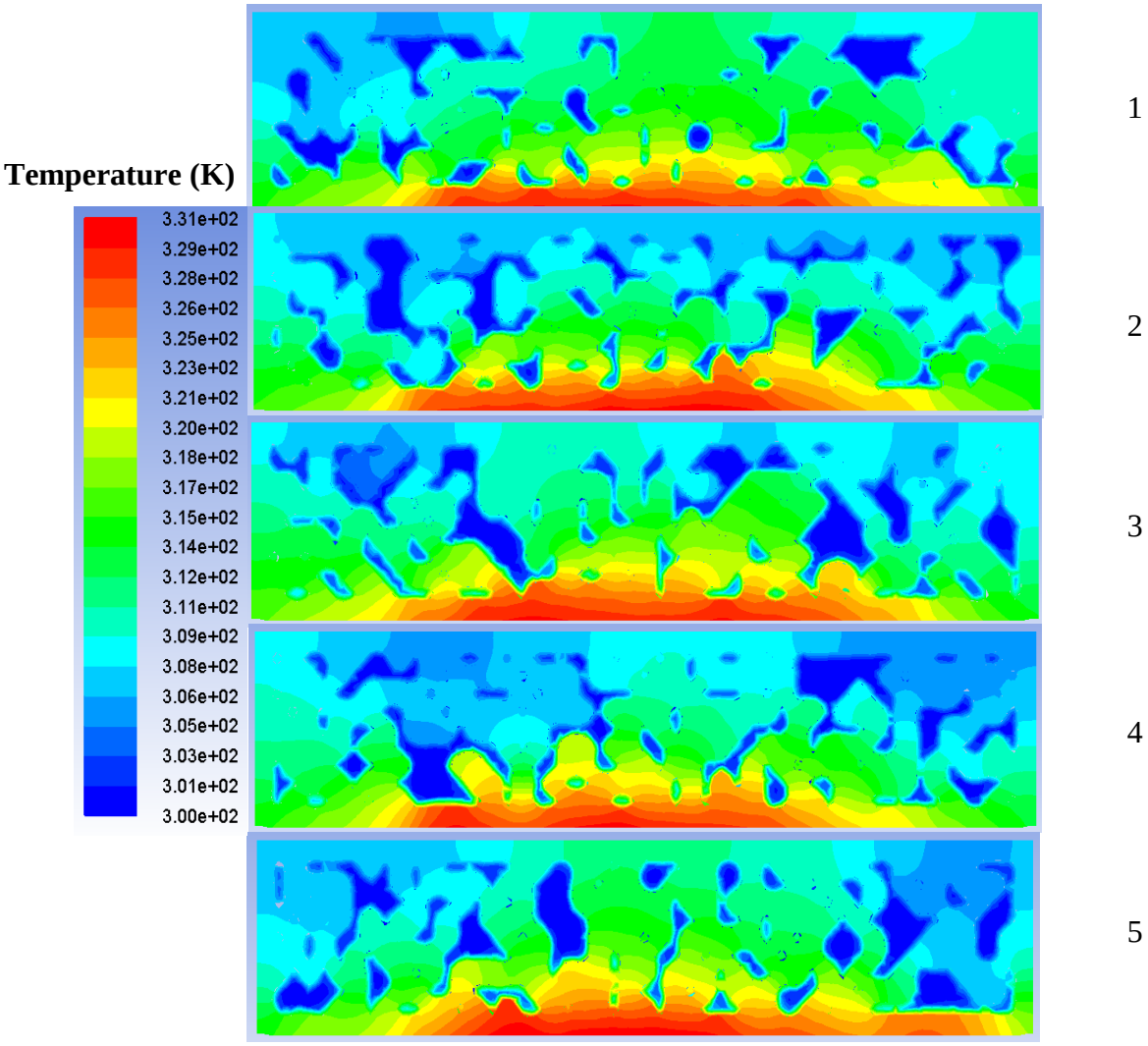

Figure 4.4: Temperature (K) profiles of the symmetric heatsink designs, located in the Z plane to find high temperature zones

4.2. Non-Symmetric (CPU) Power Map

As before in the symmetric power map analysis, 100 generations were run before analysis was stopped and the data was analyzed. Again, generation sets consist of 4 unique solutions and 1 elite solution. The first generation consists of 20 solutions, in the interest to see the effects of different initial sizes on the algorithm compared to the symmetric analysis. Unlike the symmetric power map analysis, this study utilizes a freshly generated solution set rather than a collection of strong solutions for the first generation. Therefore, the initial solution set fitness will be greater distributed than the first study. This is the default method of analysis for this algorithm, and is representative of the standard algorithm behavior.

When utilizing the power map based on the CPU power map, it was found the top five entropy generation rates were 0.01920, 0.01966, 0.02001, 0.02005 and 0.02012 W/K. This is compared to the finned heat sink at 0.02445 W/K, a reduction between 18 and 21%. The reduction in thermal resistance for the generated geometries is between 18 and 22%. Because the difference in overall power, the reduction in maximum temperature is not as significant as the symmetric map solutions, at 2.5 K. Again, this added performance comes at a cost of device mass and pressure drop compared to the finned design (at 46.76 g and 59.34 Pa). The solutions have a mass of 68.83, 70.10, 67.88, 66.95 and 67.87 grams. This is an increase in mass from 45.16 to 49.93%. As mentioned earlier, the algorithm was not optimizing for mass, so the increases are an unfortunate side effect. The top five generated designs have a pressure drop of 328.3, 395.2, 336.9, 289.1 and 302.6 Pa respectively. This is a respective increase of 453, 566, 468, 387 and 410%. As discussed before, this is a noticeable increase compared to the traditional finned system, and thermodynamic improvement from an entropy generation perspective must be weighed against practical issues such as pumping power.

The generated geometry of the top five results are displayed in Figure 4.5, along with the corresponding power map of the problem. Compared to the power map in the first analysis, the overall power levels are more uniform. As a result, we expect to see a more even distribution of mass throughout the heat sink. Again, the solutions were discretized into 5 subsections, and the average fill percentage determined from each as shown in Figure 4.6. Unlike the symmetric map, there is no obvious locational bias, with most solution sections being in the 50-55% range. Section 3 may be an exception, with most of the solutions remaining in the 55-60% range. An interesting note is the percentage fill compared to the symmetric solutions seems to be slightly higher. This can be confirmed by checking the total fill density of the devices, at 55.33, 56.24, 54.42, 52.61 and 55.33% respective to the solutions. Compared to the symmetric solutions, this a consistent increase of approximately 3-5%. This may be an effect of more evenly distributed and consistent voids in the solutions versus those in the symmetric systems.

More illustrative results can be drawn from reviewing the temperature distributions presented in Figure 4.7. Compared to the symmetric power map, temperature was distributed more evenly, with the exception of the left most edge. The voids resulting from diagonal bits continue to prove ineffective. As discussed previously, the temperature distribution indicates they offer little assistance in flow, resulting in minimal reduction in pressure drop and loss of material to dissipate energy from the surface. As mentioned when viewing the fill densities, there seems to be a more even distribution of voids in the non-symmetric solutions as a result of the relatively uniform power map. This can be confirmed by viewing the temperature distribution maps that more fluid flow occurs around the edge of the symmetric map solutions. Again, this is an effect of the power map distribution, with lower power areas enabling greater fluid flow given the lower necessary power dissipation.

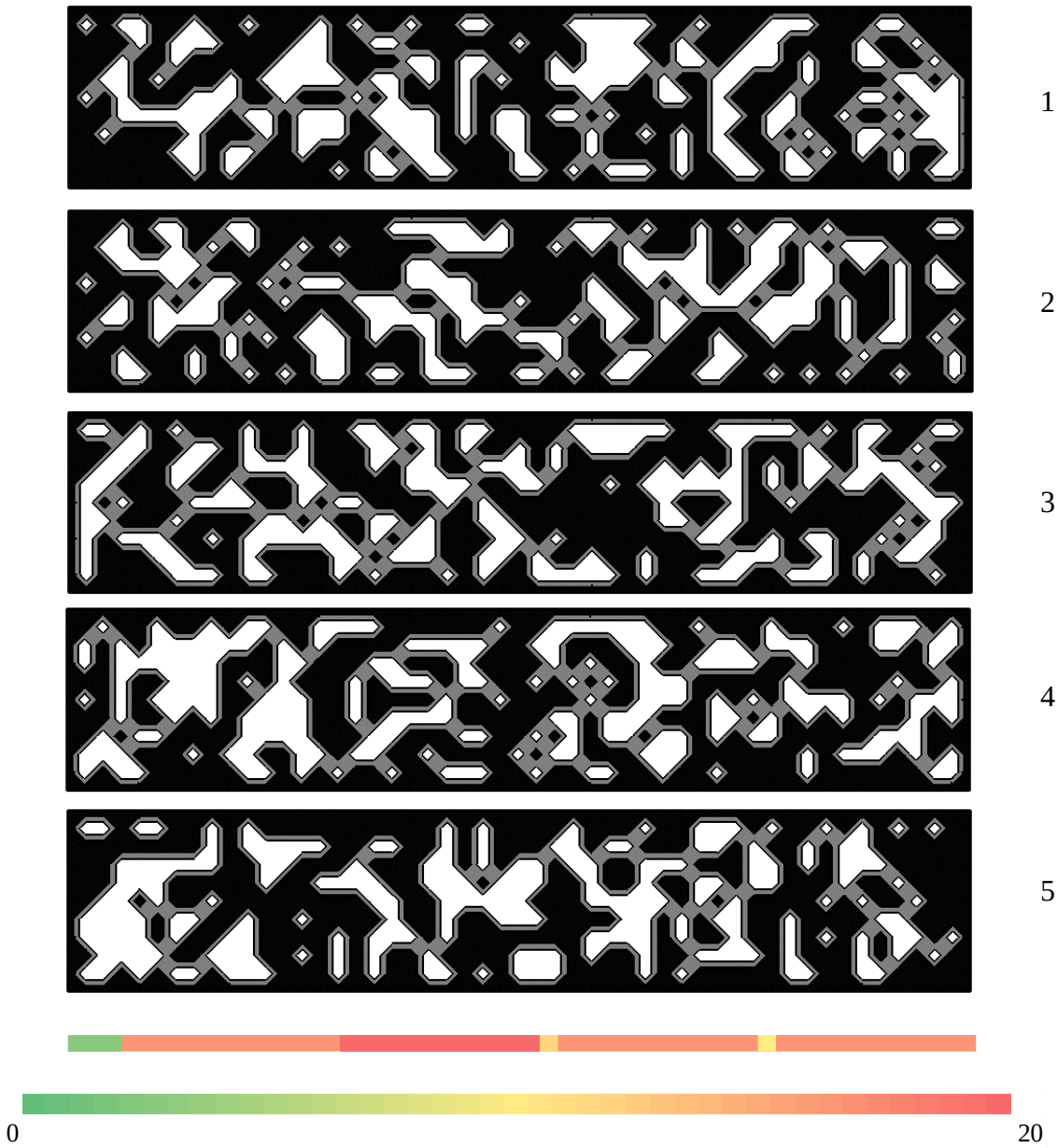

Figure 4.5: Contour figures of the top five solutions providing the lowest entropy generation rate utilizing the non-symmetric power profile. The power profile is provided below, measured in W

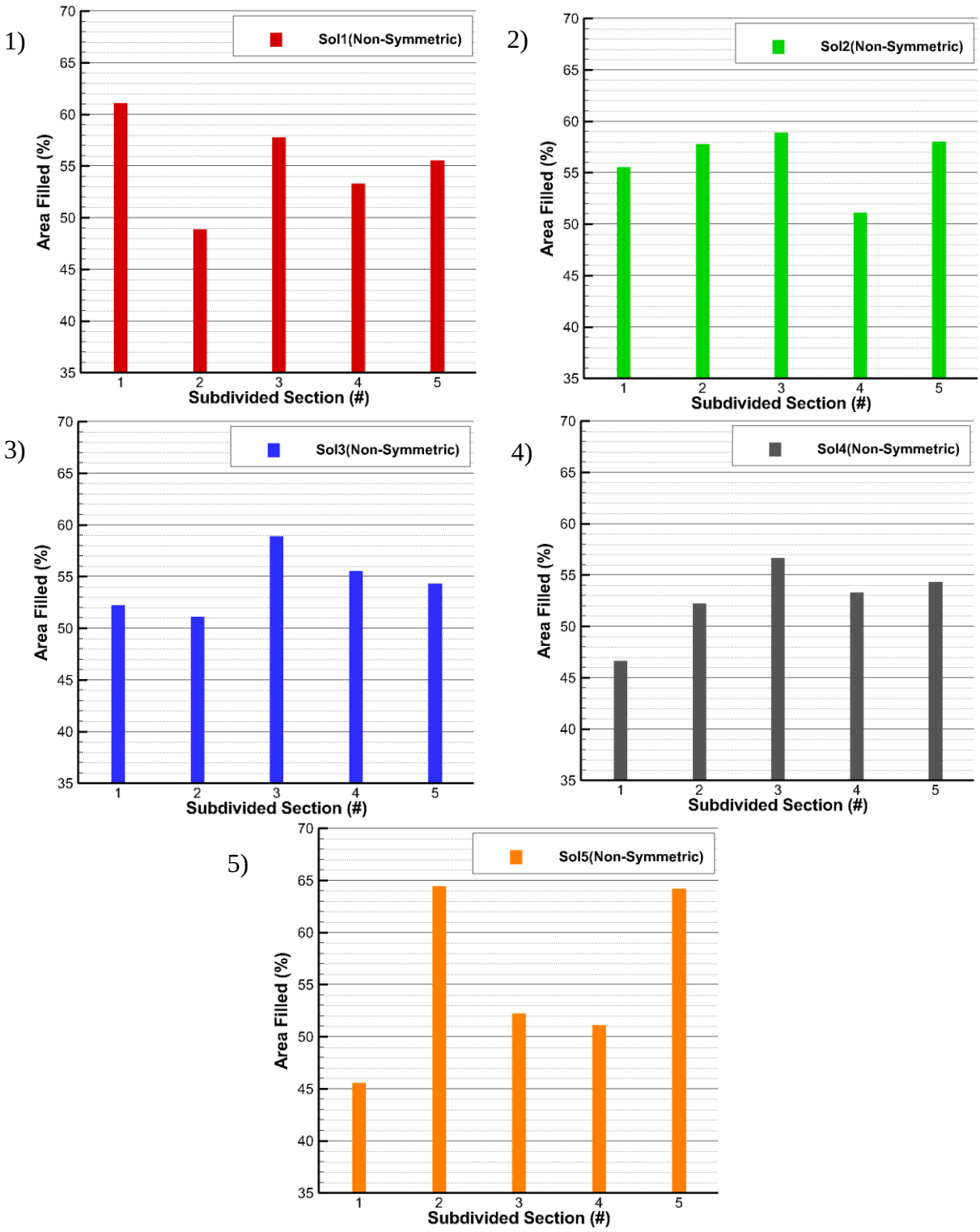

Figure 4.6: Fill densities of the subsections of the non-symmetric solutions. A) Solution 1, B) Solution 2, C) Solution 3, D) Solution 4, E) Solution 5

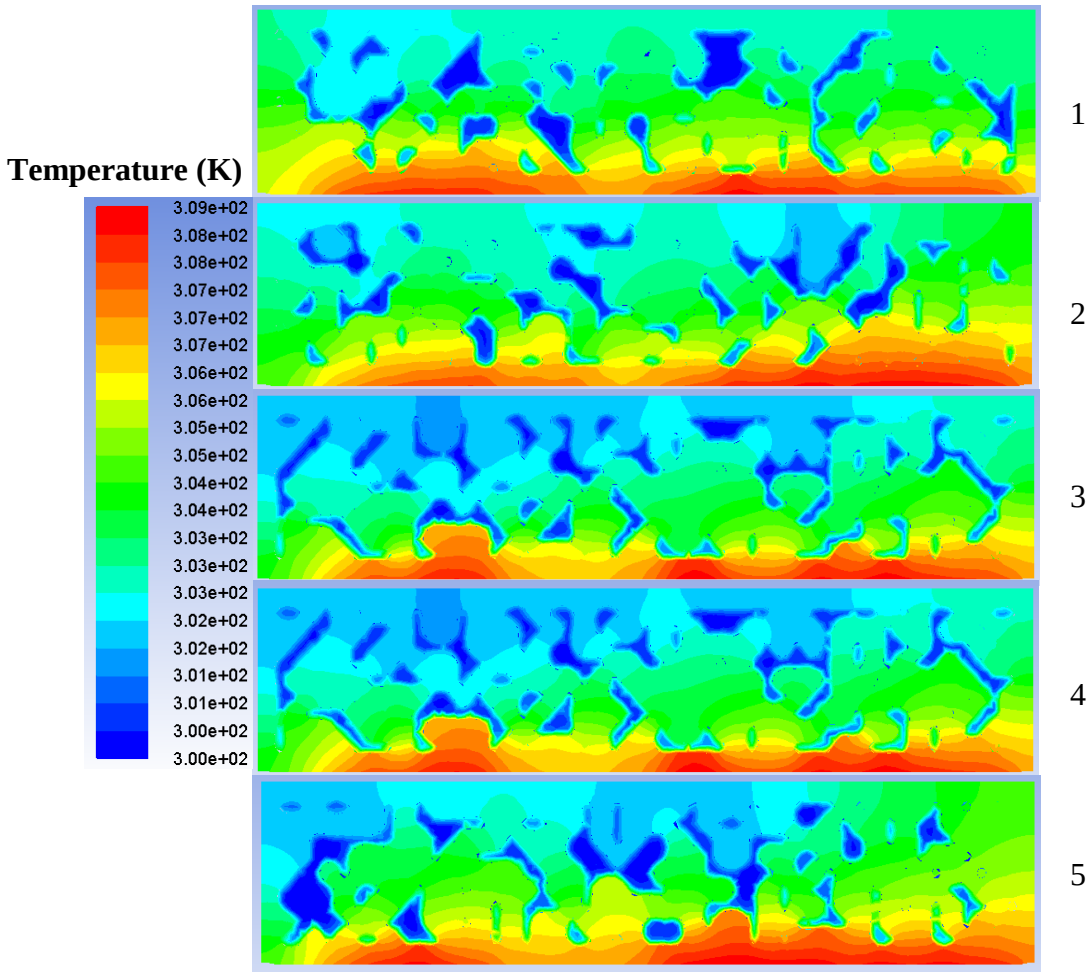

Figure 4.7: Temperature (K) profiles of the non-symmetric heatsink designs, located to find high temperature zones

4.3. Effectiveness of Algorithm

To see the effects of the generational advancement on the entropy generation rate of the solutions, the results for all generations are plotted in Figure 4.8 and 4.9 for the symmetric and non-symmetric power cases, respectively. To get a better indicator of the effects of the algorithm on solution generation, the elite solution of each generation set is excluded from this data. This is because the elite solutions will always provide the best available solution, and do not accurately represent the behavior of the algorithm. Among the first things to make note of, is that the initial population has limited effect on the generation, when comparing between the two analyses. The solution converges faster, but the best solution is not found any earlier in the analysis. The average and minimum value curve for generation using the symmetric power map is displayed in Figure 4.9, with the non-symmetric results being displayed in Figure 4.10.

Interestingly, when the entropy generation for all geometries within a generation are averaged, a somewhat random oscillating signal occurs over the generations. Meaning the results shift between achieving lower entropy generation rate values before rising to a higher average, and repeats every few generations. It is possible this is not purely an effect of the algorithm, but as a result of the bit-array and the lack of an immediate relation between a bit and the bit's effect on the solution. In other words, in most algorithms, a change of a variable has direct and noticeable impact on the results of the solution. However, in a bit-array, the addition or removal of a bit does not have a defined impact result on their own. Rather, the final result is dependent on larger groups of bits and their location in the system, complicating the solution. In the end, the ideal best solutions tend to occur randomly during the generations. It is possible that better solutions may appear with further analysis, but the computational cost must be balanced with the available results, and strong solutions were already determined within the generational count.

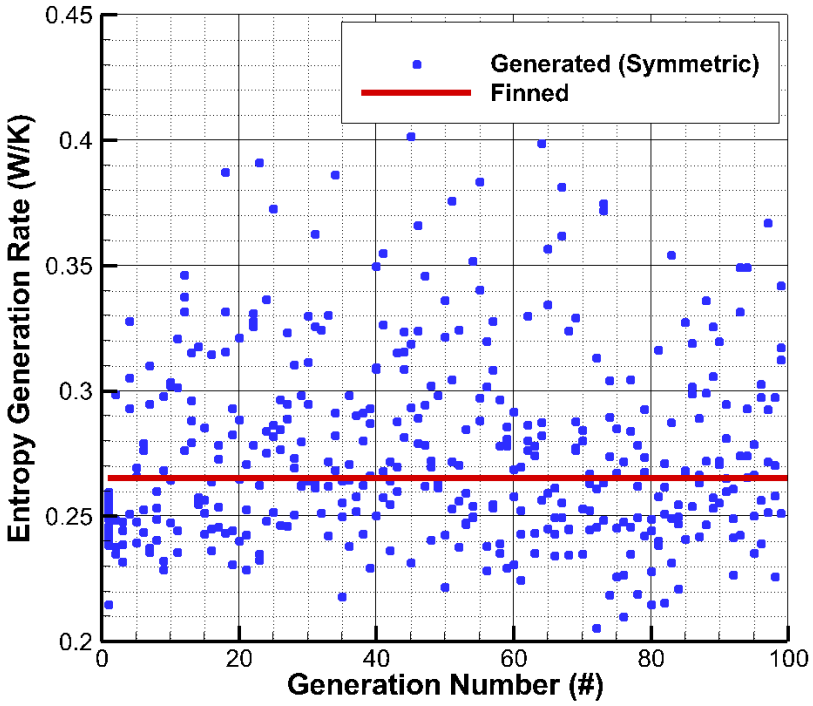

Figure 4.8: Entropy generation results for the analysis using a symmetric power map

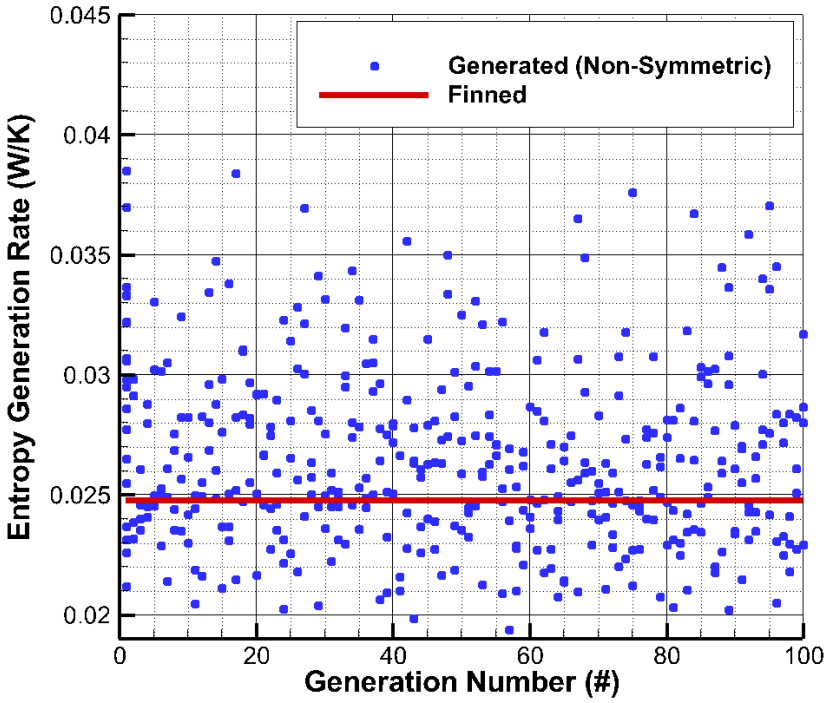

Figure 4.9: Entropy generation results for the analysis using a non- symmetric power map

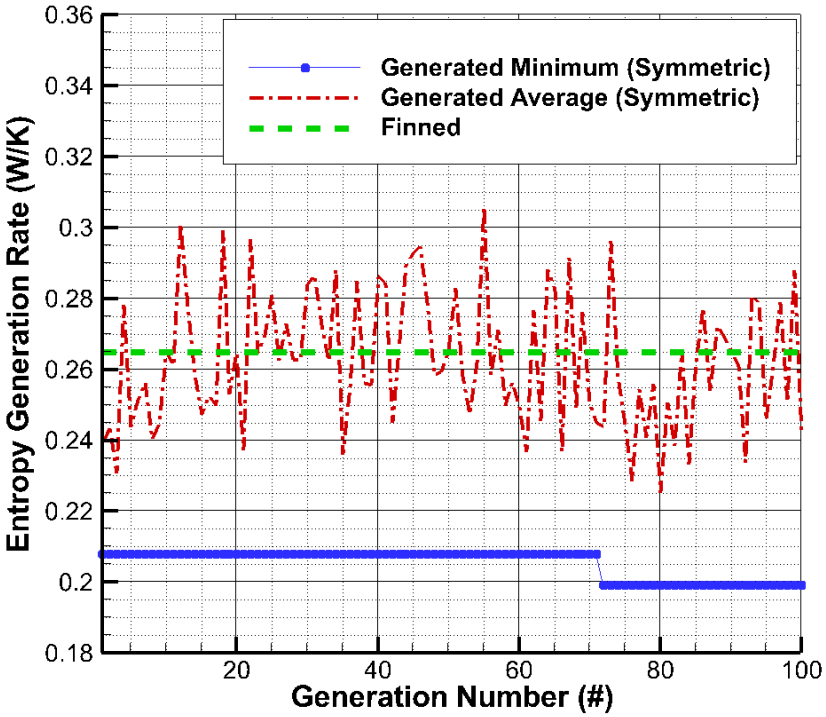

Figure 4.10: Entropy generation rate plot with averaging curve applied across the symmetric solution set

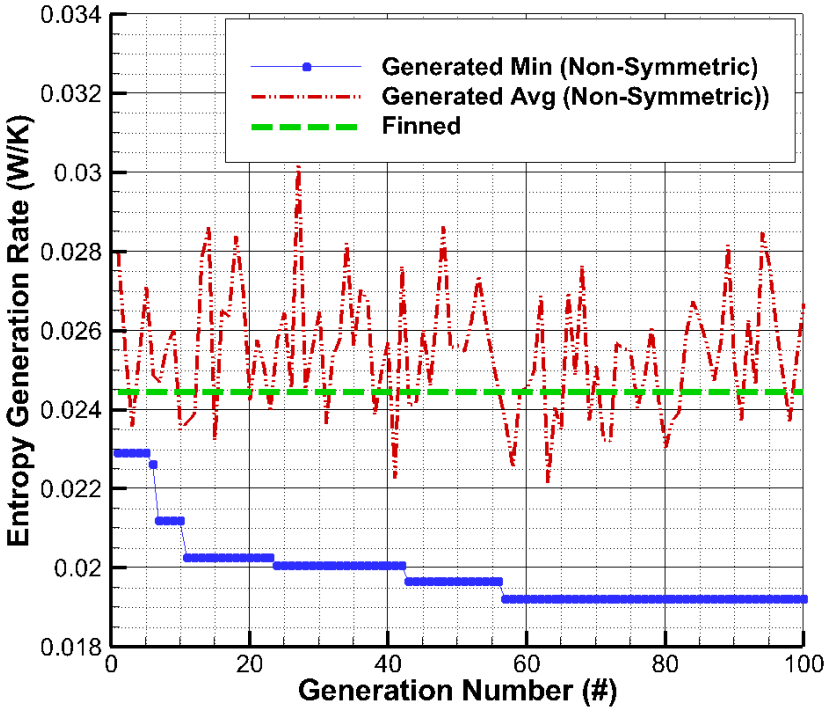

Figure 4.11: Entropy generation rate plot with averaging curve applied across the non-symmetric solution set

While results from the solution are similar to other results obtained from single-objective algorithms, it does not seem to have the effectiveness of the multi-objective algorithms found in literature. Most studies report the best possible solution in each generation, including the elite values. To provide a comparison, the best possible solution for each generation are also plotted in Figure 4.10 for the symmetric solution and Figure 4.11 for the non-symmetric. The most direct comparison available is provided by Gagne and Andersen (2010), shown in Figure 4.12, who provide both average and optimal results across their generations. When observing the average system results, a similar type of sinusoidal function occurs, indicating that the behavior encountered in this research is not uncommon. The behavior of the optimum solution across the generations is essentially logarithmic, and is clearly seen in both the second analysis research and other papers. Note that the logarithmic effect is not seen in the first analysis, as a result of the strong initial solutions. These other papers include Wu et al. (2016), Wang and Tai (2005) and Park (2010), who all see a logarithmic effect when observing the optimum functions. The vast majority of papers utilizing genetic algorithms tend to focus on the Pareto front results, which will be discussed later.

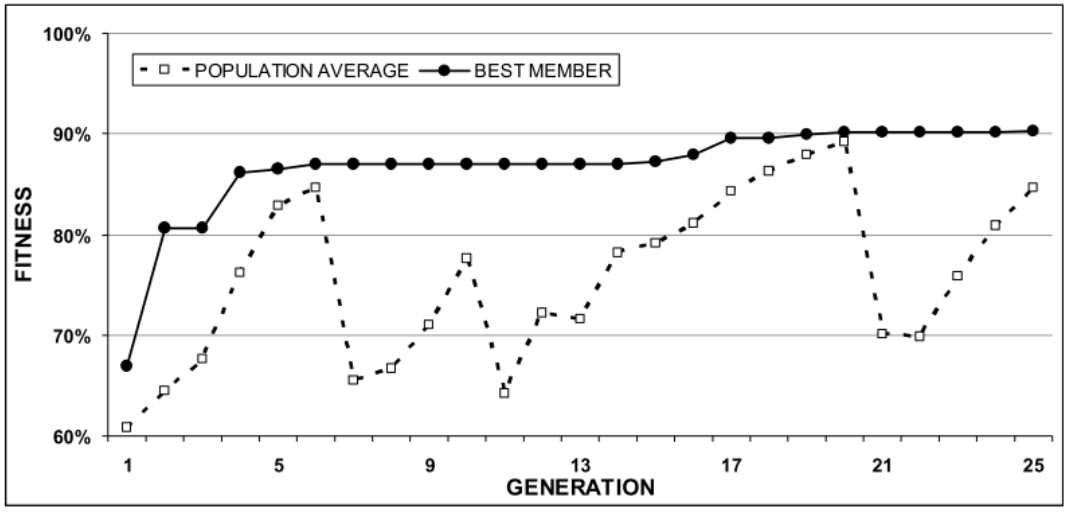

Figure 4.12: Generational results of a micro-genetic algorithm in Gagne and Andersen (2010)

While the results from the algorithm are consistent with literature, it seems further refinement could improve upon convergence and results. Ideally, another component should be added to the algorithm to direct the solution. This would be similar to Tai et al. (2005), who define a “skeleton” around which a bit-array controlled “flesh” is utilized for algorithm manipulation. However, it would be difficult to implement a defined skeletal structure around an unknown power in the system. A more logical method would be to define the ideal structures as they generate. There are a few possible methods that could be utilized to accomplish this, the basis of which is that the ideal structures need more influence on the generated solutions. One method would be structure identification, similar to the block crossovers discussed by Kane and Schoenauer (1996). In this case, the bit-array space would be subdivided into larger sections, which would allow more information from the solution set to be carried over to the new generation and potentially result in more consistent generational results. A second method would be to reintroduce the normalized values of the base power into the algorithm rather than just as part of the initialization. Modification to the crossover and mutation rates could be correlated to these normalized values close to the power surface. The hope would be this would add weighting toward the ideal solutions throughout the generations. A third, more complex methodology, would be creating evolving structural recognition and add weighting requirements to the algorithm to direct new generations to follow this structure. To provide an example, this would require another external memory source to store this base structure. This structure would be based on the best solution currently generated. When a superior solution is generated, it is compared to the current structure. A new structure is generated based only on the solid bits shared by the two solution arrays. This structure is then utilized to guide the generation toward following it, by requiring the new solutions to match the structure by a certain percentage.

Making a system such as this work effectively would require significant effort to tune, necessitating starting with a simple problem as the initial tuning media. While further testing would be needed, these methods may provide the extra steps needed to ensure the micro-genetic algorithm works more effectively with a bit-array CFD problem.

The thermal resistance and pressure drop of each solution is shown in Figures 4.13 and 4.14. A very similar result occurs when plotting the pressure drop versus the entropy generation rate as shown in Figures 4.15 and 4.16. There is a noticeable Pareto front for both solutions, but note that the best solutions (lowest entropy generation rates) were at 261 and 328 Pa for the analysis of the symmetric and non-symmetric systems, respectively. This effectively means the Pareto front occurs at the bottom of the formed curve, indicating that the solution is most influenced by reduction in thermal resistance compared to reduction of pressure drop, as governed by the formulation of the entropy generation rate. This can be confirmed when looking at the scatter plot comparing the thermal resistance to the calculated entropy generation rate, as shown in Figure 4.17 and 4.18. Note that each analysis shows a very linear relationship between the two properties, again indicating the entropy generation rate term driving the analysis is almost purely influenced by the thermal resistance. This is further demonstrated when performing the uncertainty analysis.

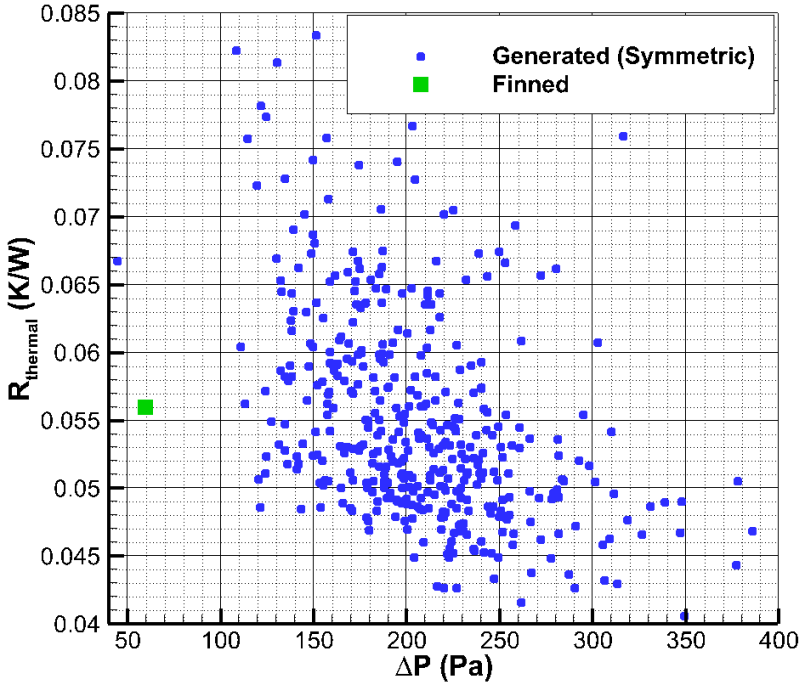

Figure 4.13: Plot of generated results across the pressure drop and thermal resistance, showing the effective Pareto front of the symmetric power map solutions

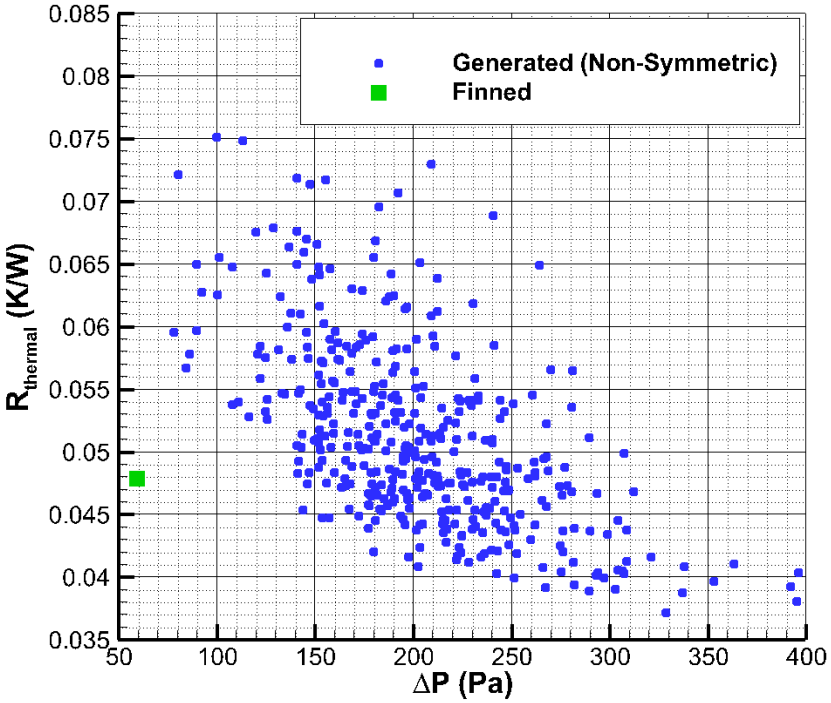

Figure 4.14: Plot of generated results across the pressure drop and thermal resistance, showing the effective Pareto front of the non-symmetric power map solutions

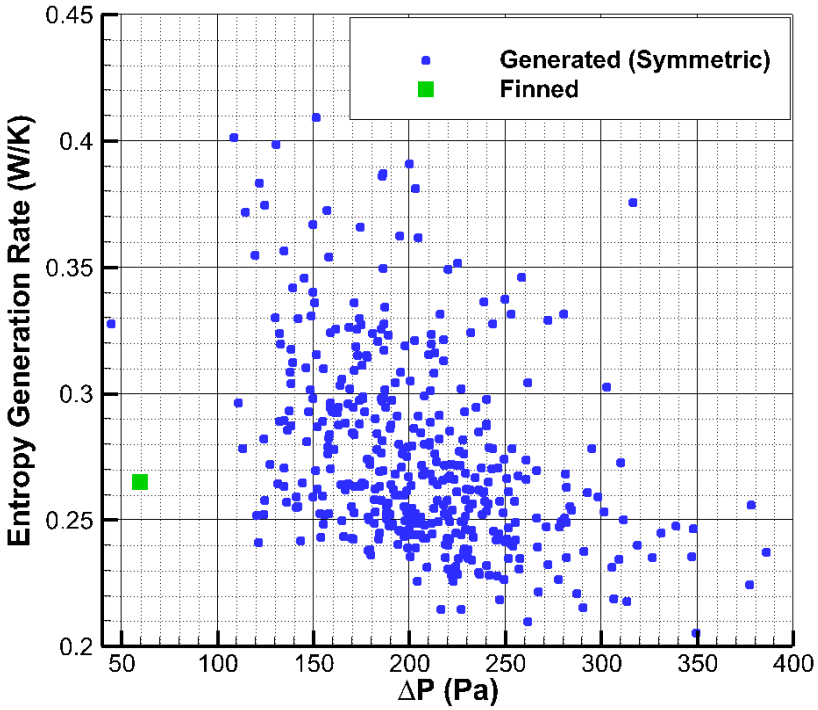

Figure 4.15: Plot of generated results across the pressure drop and entropy generation rate, showing the effective Pareto front of the symmetric power map solutions

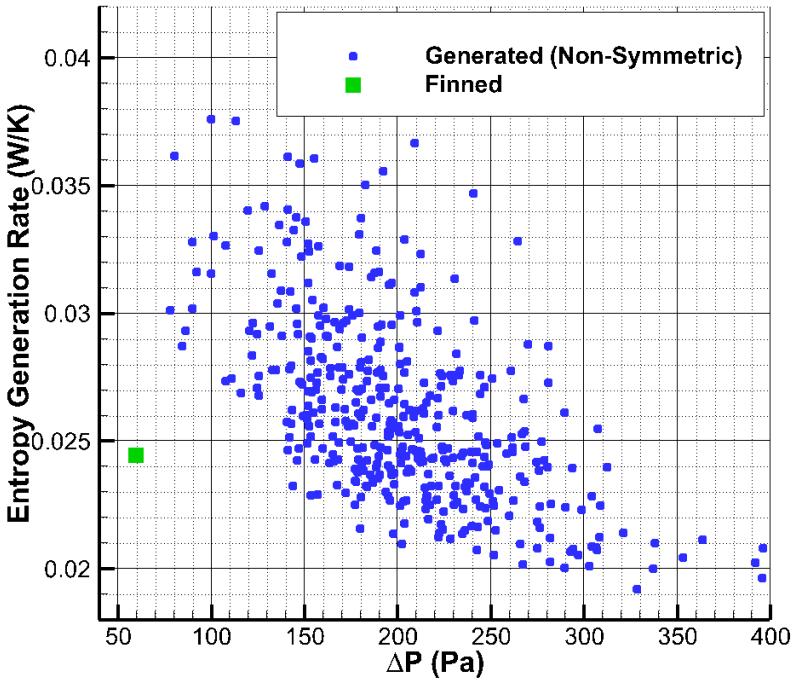

Figure 4.16: Plot of generated results across the pressure drop and entropy generation rate, showing the effective Pareto front of the non-symmetric power map solutions

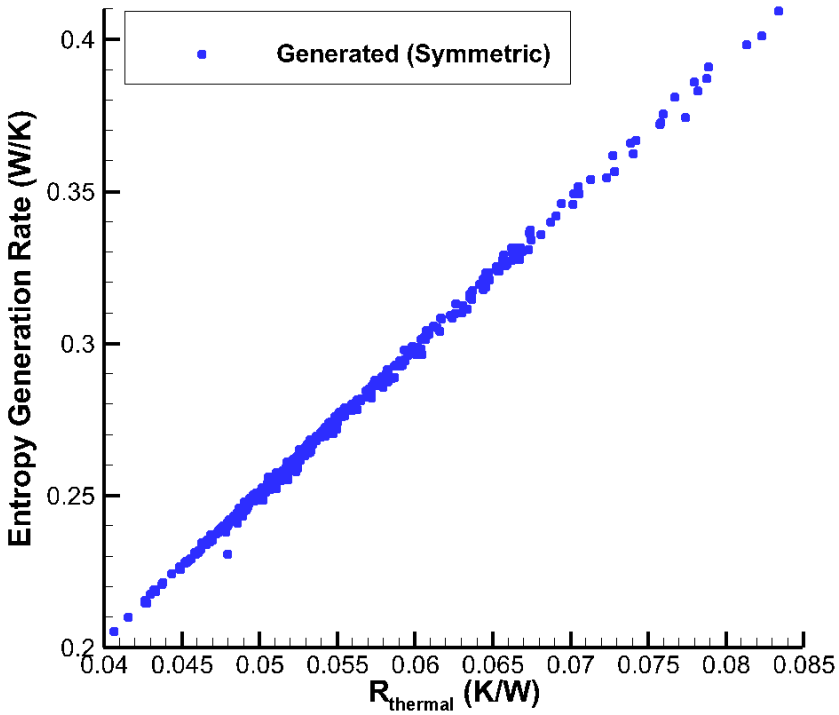

Figure 4.17: Comparison of the thermal resistance and entropy generation rate for the symmetric power map analyses, showing a linear relation.

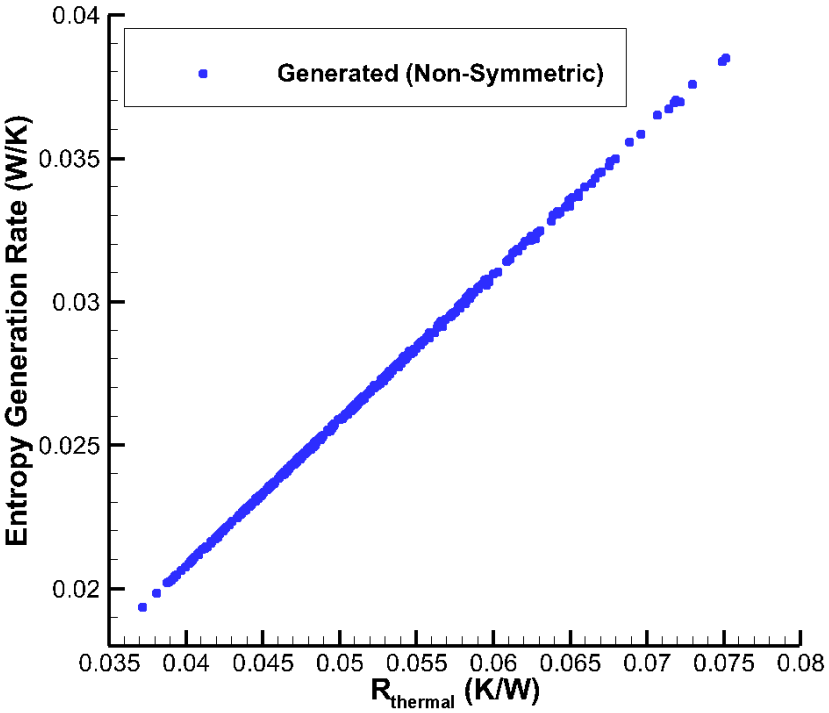

Figure 4.18: Comparison of the thermal resistance and entropy generation rate for the non-symmetric power map analyses, showing a linear relation.

4.4. Uncertainty Analysis

Uncertainty in computational fluid dynamics is comprised of four elements of acknowledged uncertainty: physical approximation, computer round off, iterative convergence and discretization (Slater, 2008). Physical approximation is an uncertainty in the choice of the system model and any simplification done for analysis. In this analysis, the largest model uncertainty comes from the choice of the Spalart-Allmaras turbulent flow model over the laminar model. The average difference between the two models is 3.44% for the pressure drop (in Table 3.3), with a maximum of 5.3%. However, based on testing results shown in Section 4.3, this uncertainty in pressured drop should have minimal effect on the entropy generation rate result. Additional model uncertainty is added when comparing the system power. A maximum of 1% difference from the actual power value was allowed, so an uncertainty of 1% is assumed. Uncertainty due to computer round off is simply a measure of the accuracy of reporting of the results. In this study, the maximum size of round off is to four decimal places. Iterative convergence uncertainty is from a solution not converging to a final value from residuals that are still reducing in value. This uncertainty should not be occurring in the solution, as all solutions are determined after the residuals have reached their minimum. Finally, discretization uncertainty is from the chosen grid, often covered by the grid convergence index. As discussed previously and shown in Table 3.2, grid convergence is within 0.53 % of a grid of half the scale in the fluid (pressure), and 1.33% of the solid material (temperature). It is an important note that the uncertainty on the temperature is based on the delta between the flow inlet temperature and the measured temperature. These uncertainty values will be utilized to determine the total uncertainty on the entropy generation rate values.

Total uncertainty for each variable was determined utilizing the root sum of squares (RSS) for the pressure drop, maximum surface temperature, average water temperature and system power input. Round off uncertainty has negligible effect, with the resulting relative uncertainty on the order of 10^{-7}. The total relative uncertainty for pressure drop was determined utilizing the maximum model choice uncertainty, putting the total uncertainty at 5.33%. The temperature uncertainty is uniform at 1.33% for each variable. Uncertainty of the system power was based on the potential allowed uncertainty in the analysis, putting it at 1%. The variables of importance were the system pressure drop, the thermal resistance and the entropy generation rate. Six different solutions for each analysis were utilized to determine the uncertainty, with two being of optimum result, two of average result and two of the worst solutions.

Uncertainty propagation was conducted utilizing the Engineering Equation Solver (EES) (2017) program, which calculates overall uncertainty using the method described in NIST Technical Note 1297 (F-Chart Software, 2017b; Taylor & Kuyatt, 1994). Complete results of the variables of interest are shown in Table 4.1. Total uncertainty of the pressure drop is not affected by propagation, and as such remains at the 5.33% determined in the total uncertainty. For every solution analyzed and for both power boundary conditions, the uncertainty in calculated thermal resistance and entropy generation rate were approximately 1.60-1.75%. The consistency in results indicates it can apply nearly universally to every solution in providing the maximum uncertainty. It also validates the solution results, as the effects of the uncertainty do not change the comparison of the best solutions significantly enough that they can be matched by the finned solution. An important note is that the largest contributors of uncertainty to these equations were the maximum surface temperature and system power, providing approximately 67% and 33% of the uncertainty respectively. This breakdown is provided in Figure 4.19. These results indicate

the maximum surface temperature has the largest effect on the solution results, as well as demonstrating the importance of confirming the accuracy of the input power.

Table 4.1: Results from propagation uncertainty analysis

Analysis 1	dP (Pa)	R_th (K/W)	S_gen (W/K)	dP Uncertainty (%)	R_th Uncertainty (%)	S_gen Uncertainty (%)
3.72	349.0	0.04060	0.1992	5.326	1.754	1.654
5.76	261.4	0.04155	0.2033	5.326	1.754	1.653
3.91	240.7	0.05258	0.2513	5.326	1.737	1.614
4.94	212.2	0.05306	0.2532	5.326	1.738	1.614
2.45	108.6	0.08228	0.3694	5.326	1.719	1.544
5.37	151.3	0.08340	0.3740	5.326	1.715	1.538

Analysis 2	dP (Pa)	R_th (K/W)	S_gen (W/K)	dP Uncertainty (%)	R_th Uncertainty (%)	S_gen Uncertainty (%)
2.57	328.3	0.03718	0.01937	5.326	1.759	1.708
4.43	395.2	0.03807	0.01984	5.326	1.753	1.700
2.68	220.7	0.05004	0.02592	5.326	1.748	1.678
5.88	200.4	0.05008	0.02593	5.326	1.742	1.676
4.17	112.8	0.07487	0.03838	5.326	1.723	1.626
19.1	99.77	0.07510	0.03850	5.326	1.734	1.635

Unit Settings: SI K Pa J mass deg

Variable±Uncertainty	Partial derivative	% of uncertainty
R_{th} = 0.03718±0.000654 [K/W]		
dP = 328.3±11.3 [Pa]	$\partial R_{th}/\partial dP$ = 0	0.00 %
dT_{max} = 8.818±0.1173 [K]	$\partial R_{th}/\partial dT_{max}$ = 0.004574	67.27 %
dT_{water} = 0.6881±0.009152 [K]	$\partial R_{th}/\partial dT_{water}$ = -0.004574	0.41 %
Q = 218.6±2.186 [W]	$\partial R_{th}/\partial Q$ = -0.0001701	32.32 %
S_{gen} = 0.0192±0.0003308 [W/K]		
dP = 328.3±11.3 [Pa]	$\partial S_{gen}/\partial dP$ = 1.666E-07	0.00 %
dT_{max} = 8.818±0.1173 [K]	$\partial S_{gen}/\partial dT_{max}$ = 0.002293	66.06 %
dT_{water} = 0.6881±0.009152 [K]	$\partial S_{gen}/\partial dT_{water}$ = -0.002418	0.45 %
Q = 218.6±2.186 [W]	$\partial S_{gen}/\partial Q$ = 0.00008755	33.49 %

No unit problems were detected.

Calculation time = 31 ms

Figure 4.19: Results from an error propagation analysis utilizing EES

4.5. Comparison to Literature

As this study is performed with water, it is difficult to make a direct comparison with other studies. The best avenue of comparison is to compare the individual performance of the created solutions to a traditional heat sink. The most immediate comparison would be to Bornoff and Parry (2015), with their organically grown heat sink. This comparison is interesting, as they achieved a 20% reduction in thermal resistance over the traditional finned heatsink. Results from this study showed somewhat higher thermal resistance reduction with the symmetric power map at 27%, but similar results with the non-symmetric power map at a 22% reduction. Bornoff and Parry also performed a comparison utilizing a traditional finned heatsink that did not have constraints on aspect ratio, and minimized the fin size to the same minimum feature size of the grown device. This heat sink actually performed better than their grown device. In this feature, the fins of the traditional heat sink are limited to 1 mm of thickness. The generated designs are similar. However, because of the contour effects, it is possible for the features to go to 0.875 mm. This only occurs in a few locations, though, with the majority of fin thicknesses being above 1 mm. Because of this, the results should still be directly comparable to the traditional finned devices.

Few studies have been done on development of heatsinks utilizing genetic algorithms, but Wu et al. (2016) has investigated algorithm usage to develop water paths in a heatsink block. They were able to achieve a reduction of thermal resistance by 15.4%, with an exchange in flow force being increased by 132%. While the usage of flow force is unusual, it is a reasonable comparison for changes in pressure drop. Usage of a perfectly uniform power across four specific blocks makes it a difficult comparison, as well as comparing to a system with a very

different design. This research was able establish a more effective improvement in thermal reduction (in exchange for greater pressure drop), but is again a very different problem set.

An interesting comparison can be made to Small et al. (2006), with their dimple/bump finned heatsink. Utilizing a uniform power across the base surface, their best design was capable of reducing thermal resistance by 22%, in exchange for an increase in pressure drop of only 34%. This is an impressive tradeoff, compared to results seen from the other studies and this research. Typically, it seems that small improvements in thermal resistance are met with a strong increase in pressure drop. As discussed previously, results are highly dependent on the system power as well as the problem assumptions. Still, the strong results provided by the study indicates the potential of modification to the fins within the flow path, in addition to avenues that may be explored with alternative manufacturing techniques.

5. Conclusion

In this research, design optimization approaches for heat sinks that could be produced through additive manufacturing was investigated. The baseline heat sink was based around a shape defined by Remsburg (2007), with a 50 mm × 10 mm × 50 mm volume and 1 mm thick walls. Material was aluminum, and the working fluid used was water. The frontal area of the heat sink was divided into a 49 × 9 grid, which was utilized to generate bits that could be translated to geometry. This geometry could then be analyzed with CFD using the ANSYS Workbench (2016d) software suite. Two power maps were used to influence the heat sink geometry generation, a simpler symmetric map and an non-symmetric map based on a real processor. The optimization process was guided through the use of a micro-genetic algorithm utilizing elitism, crossover and mutation operators, which ranked solutions by their entropy generation rate. Analysis took place over 100 generations for each power map solution set.

Results from analysis show good potential from the unrestricted heat sink design. The optimized heat sinks were able to decrease entropy generation rate by 25% and 21% for the symmetric and non-symmetric map respectively. Additionally, thermal resistance saw a respective decrease by 27% and 22%, but in exchange for greater system pressure drops at an increase of 487% and 453% each. This gain in pressure drop is a result of the use of the entropy generation rate as the guiding metric, which as a result of the small changes in the absolute pressure drop, biased the system toward solutions that greatly reduced the maximum temperature of the input surface.

These results are similar to other studies performing related investigations. Bornoff and Parry (2015) achieved a similar 20% reduction in thermal resistance with their grown heat sink

technique, but make no mention of changes in pressure drop. Wu *et al*. (2016) performed a study utilizing genetic algorithms to guide flow through a water block heat sink, and were able to achieve a 15% reduction in thermal resistance for a 132% increase in flow force (analogous to pressure drop). The difference in heat sink types makes a direct comparison difficult, however. With more traditional heat sink manufacturing, Small *et al*. (2006) were able to achieve a 22% reduction of thermal resistance in exchange for a 34% increase of pressure drop. This was accomplished through the addition of bumps and dimples to the fins. It is hard to say how these results would translate to the problem solved in this research without simulation. However, it does indicate that there are potential benefits with modification to fin geometry along the flow path.

5.1. Limitations

One of the most significant limitations of the methodology is the time required to analyze solutions. Each analysis has over 400 solutions to analyze for 100 generations, and the CFD calculation in this required approximately 2 hours to complete 2 solutions, barring any solver failures. This means it takes at least 400 hours to complete one analysis, not including the additional time for meshing and the generation of new solution sets. A few options are available to reduce the time requirement. The first would be to change the flow simulation model from the turbulence based Spalart-Allmaras model, to the laminar model. Uncertainty analysis demonstrated that accuracy with regards to the pressure drop has no significant effect on the entropy generation rate in the test setup, and therefore, any uncertainty gained from utilizing a laminar model will be negligible. From testing, this could reduce simulation time to less than 30 minutes per 2 solutions, less than a quarter of the time observed using the turbulent model. Computational time could be further reduced by utilizing better computational resources, as the

computer used was not ideal for CFD analysis. With better resources, all 4 analyses could be run simultaneously, in addition to reducing the time needed for each analysis. As well, the benefit of maintaining the process in ANSYS Workbench, is that there are scripting options available to automate procedures. This was not fully implemented during this study due to three limitations. First, the generation technique needs to be modified so that "floating" shapes are not generated. This would require a filter that can identify and modify or delete these shapes. Second, adjustments need to be made to the mesh sizing to prevent meshing uncertainties (typically needing a reduction in grid size). This will increase analysis time, but may be kept to a minimum with careful adjustment. Finally, uncertainties may occur during analysis, which is typically a result of exponential temperature increases in the small channels during iteration. Much of these uncertainties can be removed by modifying the code to eliminate the small openings resulting from diagonals bits. Combining all of these changes will result in a significant reduction in time for analysis. A probable combined time of generation, meshing and analysis could easily be dealt with in 1 hour. Assuming no other issues, a full analysis could be completed with 100 hours, less than a week with the program running independently.

The next limitation is the current state of additive manufacturing technology. The solutions in the study see minimum feature size of 0.875 mm, while some channels can be down to 0.75mm. This is well within the capabilities of commercial printers, which will commonly manage features down to 150 µm (SLM Solutions, 2017).While it would be possible to manufacturer the resultant designs, problems will be encountered when trying to apply the design to a mass manufactured product. It is important to remember that parts created through AM are essentially produced at a fixed cost with no economies of scale. As discussed by Thomas and Gilbert (2014), there is a point in the number of manufactured parts where the economies of

scale seen in traditional manufacturing techniques overcomes the fixed costs utilized by AM. Because of the necessary usage of metal printing techniques, which are uncommon and utilize costly base materials, prices will be greater than typical plastic printing processes. Further, the number of heat sinks that will be manufactured will be on the order of thousands. Given the cost associated with the necessary additive manufacturing systems, economies of scale will quickly eliminate cost benefits of AM within a small number of units. For AM heat sinks to become commercially viable, the manufacturing technology needs additional time to advance, and allow the fixed costs to be reduced.

Finally, the end goal of heat sink design utilizing additive manufacturing would be the creation of full three-dimensional structures throughout the heat sink volume. There would be a significant number of design benefits, allowing controlled management of both fluid flow and temperature reduction based on 2D power maps. It is possible that an even greater reduction in thermal resistance could result from this generation method, while having a minimal impact on pressure drop. Computational requirements remain the hurdle to expanding the methodology to 3D space. The number of data points to determine, as well as the significant increase in the number of complex surfaces, means that complete analysis cycles will take significant time. As a reference point, the finest mesh in the grid convergence study took approximately 2 hours to complete on the 2D generated geometry. With 3D generated geometry, it is possible this step could extend to a time period on the order of a days or weeks, even with better computational equipment. Similar results will occur with CFD analysis, which could easily extend to a magnitude of weeks or beyond. Combined with the number of analyses that must be run, the process is not currently reasonable for extension to 3D generated devices.

5.2. Future Research

With a promising baseline established, there are a few options available to expand the research. While this research was based on developing a heat sink by modifying the geometry in the frontal X-Y plane, there is also potential of modification along the flow path in the X-Z plane. Little change is needed to adapt the current technique to this analysis, and the results would provide a comparison to if there is benefit of modifying heat sinks in different planes. Another area that can be expanded upon is that of the optimization algorithm. In this analysis, a multi-objective problem was simplified to that of a single-objective problem by the use of entropy minimization. Expansion of the methods back toward a true multi-object algorithm would allow greater design flexibility, allowing designers to pick and choose what their optimum goals are. Finally, further optimization can be made to the methodology to enable it to become faster and easier to use. This includes steps such as automation of ANSYS Workbench through scripts, as well as the automation of the new solution generation process. Combined with small changes such as the elimination of small gaps resulting from diagonal bits, and modification of the analysis model to utilize laminar flow, uncertainties could be reduced and analysis time greatly improved. The combination of further research and technology advancement could allow the technique to become a viable option for use within industry.

References

ANSYS. (2006). *ANSYS Fluent 6.3 User Guide*. Retrieved from https://www.sharcnet.ca/Software/Fluent6/html/ug/node1.htm

ANSYS. (2016a). DesignModeler.

ANSYS. (2016b). Fluent.

ANSYS. (2016c). Meshing.

ANSYS. (2016d). Workbench.

Atzeni, E., & Salmi, A. (2012). Economics of additive manufacturing for end-usable metal parts. *International Journal of Advanced Manufacturing Technology*, *62*(9–12), 1147–1155. http://doi.org/10.1007/s00170-011-3878-1

Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. *Icml*, 1–11. http://doi.org/10.1.1.44.5424

Bauer, T., Dawson, K., Spierings, A. B., & Wegener, K. (2013). Microstructure and mechanical characterisation of SLM processed Haynes® 230. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699. http://doi.org/10.1017/CBO9781107415324.004

Bejan, A. (1982). *Entropy Generation Minimization*.

Bonded Fin Image. (2017). Retrieved from http://thermocoolcorp.com/project/bonded-fins/]

Bornoff, R., & Parry, J. (2015). An additive design heatsink geometry topology identification and optimisation algorithm. *Annual IEEE Semiconductor Thermal Measurement and Management Symposium*, *2015–April*, 303–308. http://doi.org/10.1109/SEMI-THERM.2015.7100177

Bornoff, R., Wilson, J., Parry, J., Court, H., & Uk, S. (2016). Subtractive Design : A Novel Approach to Heatsink Improvement Mentor Graphics , Mechanical Analysis Division. *Semi-Therm Symposium*.

Chapman, C. D., Saitou, K., & Jakiela, M. J. (1994). Genetic Algorithms as an Approach to Configuration and Topology Design. *Journal of Mechanical Design*, *116*(4), 1005. http://doi.org/10.1115/1.2919480

Coello, C. A. C., & Pulido, G. T. (2001). A Micro-Genetic Algorithm for Multiobjective Optimization, 126–140.

Corana, A., & C. Martini and S. Ridella, M. and. (1989). Corrigenda: "Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing' Algorithm." *ACM Transactions on Mathematical Software*, *15*(3), 287. http://doi.org/10.1145/66888.356281

Culham, R., & Muzychka, Y. (2001). Optimization of plate-fin heat sinks using entropy

generation minimization. *IEEE Transactions on Components and Packaging Technologies*, *24*(2), 159–165.

Czyzak, P., & Jaszkiewicz, A. (1998). Pareto Simulated Annealing--A Metaheuristic Technique for Multiple-Objective Combinatorial Optimization. *Journal of Multi-Criteria Decision Analysis*, *7*(1), 34–47. http://doi.org/10.1002/(SICI)1099-1360(199801)7:1<34::AID-MCDA161>3.0.CO;2-6

Deb, K., & Agrawal, S. (1999). Understanding interactions among genetic algorithm parameters. *Foundations of Genetic Algorithms V, San Mateo, CA: Morgan Kauffman*, 265–286. http://doi.org/citeulike-article-id:4372866

Deb, K., & Deb, D. (2014). Analyzing Mutation Schemes for Real-Parameter Genetic Algorithms. *International Journal of Artificial Intelligence and Soft Computing*, *4*(1), 1–28. http://doi.org/10.1504/IJAISC.2014.059280

Eberhart, R. C., & Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization. *Evolutionary Programming VII*, 611–616. http://doi.org/10.1007/BFb0040812

Foli, K., Okabe, T., Olhofer, M., Jin, Y., & Sendhoff, B. (2006). Optimization of micro heat exchanger: CFD, analytical approach and multi-objective evolutionary algorithms. *International Journal of Heat and Mass Transfer*, *49*(5–6), 1090–1099. http://doi.org/10.1016/j.ijheatmasstransfer.2005.08.032

Gagne, J. M. L., & Andersen, M. (2010). MULTI-OBJECTIVE FACADE OPTIMIZATION FOR DAYLIGHTING DESIGN USING A GENETIC ALGORITHM. *IBPSA*, *9*, 110–117.

Gen, M., & Cheng, R. (1999). *Genetic Algorithms and Engineering Optimization*. *Engineering*. http://doi.org/10.1002/9780470172261

Ghiaasiaan, S. M. (2008). *Two-Phase Flow, Boiling, and Condensation: In Conventional and Miniature Systems*. http://doi.org/10.1017/CBO9780511619410

Gillot, C., Bricard, A., & Schaeffer, C. (2000). Single- and two-phase heat exchangers for power electronic components. *International Journal of Thermal Sciences*, *39*, 826–832. http://doi.org/10.1016/S1290-0729(00)00278-7

Greene, C. (2014). C2xyz contour matrix to coordinates. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/43162-c2xyz-contour-matrix-to-coordinates

Haskell, M., & Quesnel, N. (2017). How Do Heat Sink Materials Impact Performance. Retrieved February 8, 2017, from https://www.qats.com/cms/2017/04/19/how-do-heat-sink-materials-impact-performance/

Hilbert, R., Janiga, G., Baron, R., & Thévenin, D. (2006). Multi-objective shape optimization of a heat exchanger using parallel genetic algorithms. *International Journal of Heat and Mass Transfer*, *49*(15–16), 2567–2577. http://doi.org/10.1016/j.ijheatmasstransfer.2005.12.015

Hopkinson, N., & Dickens, P. (2003). Analysis of rapid manufacturing - Using layer

manufacturing processes for production. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *217*(1), 31–39. http://doi.org/10.1243/095440603762554596

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, *1*, 82–87. http://doi.org/10.1109/ICEC.1994.350037

Houck, C. R., & Kay, M. G. (2008). A Genetic Algorithm for Function Optimization : A Matlab Implementation. *Ncsuie Tr*, *95*(919), 1–14. http://doi.org/10.1109/ISDEA.2010.135

Kandlikar, S. G. (2005). High Flux Heat Removal with Microchannels—A Roadmap of Challenges and Opportunities. *Heat Transfer Engineering*, *26*(8), 5–14. http://doi.org/10.1080/01457630591003655

Kane, C., & Schoenauer, M. (1996). Topological optimum design using genetic algorithms. *Control and Cybernetics*, *25*(5), 1059–1087.

Kang Tai, Shengyin Wang, S. A. and J. P. (2005). Structural topology optimization using a genetic algorithm with a morphological geometric representation scheme. *Structural and Multidisciplinary Optimization*, *30*(2), 113–127. http://doi.org/10.1007/s00158-004-0504-y

Khan, W. a., Culham, J. R., & Yovanovich, M. M. (2008). Optimization of Pin-Fin Heat Sinks in Bypass Flow Using Entropy Generation Minimization Method. *Journal of Electronic Packaging*, *130*(3), 31010. http://doi.org/10.1115/1.2965209

Lajevardi, B., Leith, S. D., King, D. A., Paul, B. K., Engr, M., & Northwest, P. (2011). Arrayed Microchannel Manufacturing Costs for an Auxiliary Power Unit Heat Exchanger, (1).

Lee, S. (1995). Optimum design and selection of heat sinks. *Proceedings of 1995 IEEE/CPMT 11th Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, *18*(4), 48–54. http://doi.org/10.1109/STHERM.1995.512051

Luo, X., Xiong, W., Cheng, T., & Liu, S. (2009). Design and optimization of horizontally-located plate fin heat sink for high power LED street lamps. *Proceedings - Electronic Components and Technology Conference*, 854–859. http://doi.org/10.1109/ECTC.2009.5074112

MathWorks. (2015). MATLAB.

Murr, L. E., Martinez, E., Amato, K. N., Gaytan, S. M., Hernandez, J., Ramirez, D. A., … Wicker, R. B. (2012). Fabrication of metal and alloy components by additive manufacturing: Examples of 3D materials science. *Journal of Materials Research and Technology*, *1*(1), 42–54. http://doi.org/10.1016/S2238-7854(12)70009-1

Pandey, V., Mourelatos, Z. P., & Nikolaidis, E. (2013). Limitations of Pareto Front in Design Under Uncertainty and Their Reconciliation. *Journal of Mechanical Design*, *135*(7), 71010. http://doi.org/10.1115/1.4024224

Park, S. W. (2010). Optimization of combustion chamber geometry for stoichiometric diesel combustion using a micro genetic algorithm. *Fuel Processing Technology*, *91*(11), 1742–1752. http://doi.org/10.1016/j.fuproc.2010.07.015

Paul, B. K., & Peterson, R. B. (1999). Microlamination for microtechnology-based energy, chemical, and biological systems. *ASME International Mechanical Engineering Congress and Exposition, Nashville, Tennessee, AES*, *39*, 45.

Pham, T. B., Hoyle, C., & Bay, B. (2014). Robust Topology Optimization Under Random Load Locations. In *ASME 2014 International Mechanical Engineering Congress and Exposition* (pp. 1–8). http://doi.org/10.1115/IMECE2014-36824

Remsburg, R. (2007). Nonlinear fin patterns keep cold plates cooler. *Power Electronics Technology*, *33*(2), 22–27. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-33947281868&partnerID=40&md5=b3bc09102bb01f2d649dcb5d75a8de84

Ritzer, T. M., & Lau, P. G. (1994). Economic optimization of heat sink design. *AIP Conference Proceedings*, *316*, 177–180. http://doi.org/10.1063/1.46788

Slater, J. (2008). Uncertainty and Error in CFD Simulations. Retrieved January 1, 2017, from https://www.grc.nasa.gov/www/wind/valid/tutorial/errors.html

Small, E., Sadeghipour, S. M., & Asheghi, M. (2006). Heat Sinks With Enhanced Heat Transfer Capability for Electronic Cooling Applications. *Journal of Electronic Packaging*, *128*(3), 285. http://doi.org/10.1115/1.2229230

Software, F.-C. (2017a). Engineering Equation Solver.

Software, F.-C. (2017b). Uncertainty Propagation. Retrieved from http://www.fchart.com/ees/eeshelp/1mvc0zd.htm

Solutions, S. (2017). Selective Laser Melting Machine SLM 280 2.0. Retrieved from https://slm-solutions.com/products/machines/selective-laser-melting-machine-slm-280-20

Source, C. (2017). Heatsinks. Retrieved from http://www.coolingsource.com/heatsinks/

Stafford, J., Walsh, E., Egan, V., Walsh, P., & Muzychka, Y. S. (2010). A Novel Approach to Low Profile Heat Sink Design. *Journal of Heat Transfer*, *132*(9), 91401. http://doi.org/10.1115/1.4001626

Taylor, B. N., & Kuyatt, C. E. (1994). Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results. *NIST Technical Note*, *1297*, 20. http://doi.org/10.6028/NIST.TN.1900

Thomas, D., & Gilbert, S. (2014). Costs and Cost Effectiveness of Additive Manufacturing - A Literature Review and Discussion. *NIST Special Publication*, *1176*, 1–77. http://doi.org/10.6028/NIST.SP.1176

Tuckerman, D. B., & Pease, R. F. W. (1981). High-performance heat sinking for VLSI. *IEEE*

Electron Device Letters, *2*(5), 126–129. http://doi.org/10.1109/EDL.1981.25367

Wang, S. Y., & Tai, K. (2005). Structural topology design optimization using Genetic Algorithms with a bit-array representation. *Computer Methods in Applied Mechanics and Engineering*, *194*(36–38), 3749–3770. http://doi.org/10.1016/j.cma.2004.09.003

Wei, X., & Joshi, Y. (2002). Optimization study of stacked micro-channel heat sinks for micro-electronic cooling. *InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, ITHERM*, *2002–Janua*(1), 441–448. http://doi.org/10.1109/ITHERM.2002.1012490

Wong, K. V., & Hernandez, A. (2012). A Review of Additive Manufacturing. *ISRN Mechanical Engineering*, *2012*, 1–10. http://doi.org/10.5402/2012/208760

Wu, T., Ozpineci, B., & Ayers, C. (2016). Genetic algorithm design of a 3D printed heat sink. *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 3529–3536. http://doi.org/10.1109/APEC.2016.7468376

XJet 3D. (2017). Retrieved from https://xjet3d.com

Yu, S. H., Lee, K. S., & Yook, S. J. (2011). Optimum design of a radial heat sink under natural convection. *International Journal of Heat and Mass Transfer*, *54*(11–12), 2499–2505. http://doi.org/10.1016/j.ijheatmasstransfer.2011.02.012

Zielinski, K., & Laur, R. (2006). Constrained Single-Objective Optimization Using Differential Evolution, 223–230.

Zielinski, K., & Laur, R. (2006). Constrained Single-Objective Optimization Using Particle Swarm Optimization. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 443–450. http://doi.org/10.1109/CEC.2006.1688343

Appendices

Appendix A: Power Map Generation

```
clear
clc
close all

%Sets the scale and number of points in the system grid
mult=1;
base=50;
n=base*mult;

%Sets the limits for boundary regions (e.g. where each heat zone falls)
val1s=n*0.4;
val1e=n*0.6;
val2s=n*0.2;
val2e=n*0.8;

%% Calculates the location point of each node, assigns a power value based on location (W)
for i=1:n
   for j=1:n
      x(i,j)=(0.5*10^-3)/mult+(10^-3*(j-1))/mult;
      y(i,j)=-0.0011;
      z(i,j)=-((0.5*10^-3)/mult+(10^-3*(i-1))/mult);
      if i>=val1s && j>=val1s && i<=val1e && j<=val1e
         HF(i,j)=600000;
      elseif i>=val2s && j>=val2s && i<=val2e && j<=val2e
         HF(i,j)=400000;
      else
         HF(i,j)=200000;
      end
   end
end

%Finds the highest power area buy summing the total in a line of the
%array, writes vector to file
tot=sum(HF,2);
[vmax, ind]=max(tot);
vectl=HF(ind,:);
vect=vectl';
csvwrite('HF_vector.csv',vectl)

%% Converts results into vectors, sends to a csv file to plug into ANSYS profile

x_ar=reshape(x, [n^2 1]);
y_ar=reshape(y, [n^2 1]);
z_ar=reshape(z, [n^2 1]);
```

```
HF_ar=reshape(HF, [n^2 1]);

newdata=[x_ar,y_ar,z_ar,HF_ar];

csvwrite('HeatFluxProfile1.csv',newdata)
```

Appendix B: Generation Code

```
clear
clc
close all

%Establishes physical sizing, cell count and cell spacing
W_in=50;
H_in=10;
mult=1;
W_cell=W_in*mult;
H_cell=H_in*mult;
dW=W_in/W_cell;
dH=H_in/H_cell;

%Baseline for center of cells
x_L=dW/2:dW:W_in-dW/2;
y_L=dH/2:dH:H_in-dH/2;

%Imports and normalizes power line (from highest overall power)
HF=csvread('HF_vector.csv');
HF_adj=[0 HF 0];
HF_max=max(HF_adj);
HF_min=min(HF);
HF_n=(HF-HF_min)./(HF_max-HF_min);
HF_norm=[0 HF_n 0];

for k=1:2

%Sets the initial area to values of one and determines the total number of
%interior cells
A=ones([H_cell+1 W_cell+1]);
total=W_cell*H_cell;

%Probability settings for first layer generation
p_max=0.8;
p_min=0.15;


%% Creates the first layer based off of probability and the normalized power vector

for i=2
   for j=2:W_cell
      prob=p_min+(p_max-p_min)*HF_norm(j);
      rand1=rand(1);
      if rand1<=prob
```

```
            A(i,j)=0;
        end
    end
end

%%
%Sets the boundary for viewing and generation purposes
A(1,:)=0;
A(H_cell+1,:)=0;
A(:,1)=0;
A(:,W_cell+1)=0;

%Goes through and generates points. 50 percent chance of generation, relies
%on other points around to exist
flag=0;
iter=0;
int=0;
while flag==0

    iter=iter+1;
    for i=3:H_cell
        for j=2:W_cell
            rand1=rand(1);
            prob=0.5;
            if rand1<=prob && (A(i-1,j-1)==0 || A(i-1,j)==0 || A(i-1,j+1)==0)
                A(i,j)=0;
            elseif rand1<=prob && (A(i,j-1)==0 || A(i,j+1)==0)
                A(i,j)=0;
            elseif rand1<=prob && (A(i+1,j-1)==0 || A(i+1,j)==0 || A(i+1,j+1)==0)
                A(i,j)=0;
            end
        end
    end

    flag=1;

end


%%
%Prints binary generation to csv file

Array=A(2:H_cell,2:W_cell);
csvwrite(['Shape_array',num2str(k),'.csv'],Array);

%%
```

```
BW=['BWFigure',num2str(k)];
CF=['ContourFigure',num2str(k)];

%Plots the binary image of the points (note, image is mirrored from
%physical representation)
[r,c]=size(A);
imagesc((1:c)+0.5,(1:r)+0.5,A)
colormap(gray);
axis off
axis equal
whitebg
print(1,'-dpng',BW)

%Plots the contours of the resulting system
figure
[Cont1,Cont2]=contourf(A,2);
colormap(hot)
axis on
axis equal
whitebg
print(2,'-dpng',CF)

close(1)
close(2)

%%
%Utilizes exterior program to pull contour information
[x,y,z]=C2xyz(Cont1);

%Finds the length of the contour groups
iter2=0;
l1=length(x);
for i=1:l1
   if z(i)>0.5
      iter2=iter2+1;
      length_c(i)=length(x{1,i});
   end
   length_cell(i)=length(x{1,i});
end

%Finds the maximum length of the cells
maxl=max(length_c);

%Extracts the locations of the contours from the results
iter3=0;
x_loc=zeros(maxl,iter2);
```

```
y_loc=zeros(maxl,iter2);
for i=1:l1
   if z(i)>0.5
      iter3=iter3+1;
      group(iter3)=iter3;
      length_t(iter3)=length(x{1,i});
      for j=1:length_cell(i)
         x_loc(j,iter3)=(51/50)*((x{1,i}(1,j)*dW)-dW*1.5);
         y_loc(j,iter3)=(11/10)*((y{1,i}(1,j)*dH)-dH*1.5);
         point(j,iter3)=j;
      end
   end
end
z_loc=0;


%%
%Prints information into text file formatted for ANSYS

header1='#Group';
header2='Point';
header3='X-cord';
header4='Y-cord';
header5='Z-cord';

fid2=fopen(['CurveFile',num2str(k),'.txt'],'w');

fprintf(fid2, [ header1 '\t' header2 '\t' header3 '\t' header4 '\t' header5 '\r\n']);
fprintf(fid2,'#Group 1 \r\n');

for i=1:iter3
   for j=1:length_t(i)-1
      fprintf(fid2, '%1.0f \t%2.0f \t%2.2f \t%2.2f \t%1.f \r\n', group(i), point(j,i), x_loc(j,i),
y_loc(j,i), z_loc);
   end
   fprintf(fid2, '%1.0f \t0 \r\n', group(i));
   if i<iter3
      fprintf(fid2, '#Group %1.0f \r\n', group(i+1));
   end
end

fclose(fid2);

end
```

Appendix C: First Generation Optimization

```
clear
clc
close all

%% Set the generation number for naming information
genNum=2;
memLen=25;

%% Read the generational (to be memory) arrays
for i=1:memLen
   memA{i}=csvread(['Shape_array',num2str(i),'.csv']);
end
memA=memA';

%% Import memory data for generation, extract to variables
memRes=csvread('ExtMem.csv',0);
memLoc=memRes(:,1);
dR=memRes(:,2);
dP=memRes(:,3);
dS=memRes(:,4);
numM=1:length(memA);
matrixs=size(memA{1});

%% Add borders to system arrays
for i=1:memLen
   memAM{i}=zeros([matrixs(1)+2 matrixs(2)+2]);
   for j=2:matrixs(1)+1
      for k=2:matrixs(2)+1
         memAM{i}(j,k)=memA{i}(j-1,k-1);
      end
   end
end

%% Rank memory solutions
for i=1:length(dR)
  wSum(i)=dS(i);
end
cSum=[wSum;numM]';
sSum=sortrows(cSum);

%% Crossover operations
nxoKids=4;            %Sets the number of crossovers
xoKids=cell(nxoKids,1); %Establishes the number of cells
index1=1;             %Value used to guide crossover selections
```

```
%Crossover for 2 to 5
for i=1:nxoKids              %Lists for offspring 2-5
   arr1=sSum(index1,2);       %Sets the value to be used for parent 1
   arr2=sSum(index1+1,2);     %Sets the value to be used for parent 2
   xoKids{i}=memAM{arr1};     %Sets the initial array to parent 1
   for j=2:matrixs(1)+1       %Moves through grid array
      for k=2:matrixs(2)+1
         %Sum cells around current cell of parent 1
         sumCell=memAM{arr1}(j-1,k-1)+memAM{arr1}(j,k-1)+memAM{arr1}(j+1,k-
1)+memAM{arr1}(j-1,k)...
            +memAM{arr1}(j+1,k)+memAM{arr1}(j-
1,k+1)+memAM{arr1}(j,k+1)+memAM{arr1}(j+1,k+1);
         %Sets probability values for adding or deleting cell
         rand1=rand(1);
         prob1=0.5;
         prob1d=0.65;

         %Checks if to take parent 2 value, checks to make sure a point
         %can be generated or deleted in parent 1. Generate requires a
         %cell around the current cell to exist. Delete is based on the
         %sum of cells around the current
         if rand1<=prob1 && memAM{arr2}(j,k)==0
            if memAM{arr1}(j-1,k-1)==0 || memAM{arr1}(j-1,k)==0 || memAM{i}(j-1,k+1)==0
               xoKids{i}(j,k)=0;
            elseif memAM{arr1}(j,k-1)==0 || memAM{arr1}(j,k+1)==0
               xoKids{i}(j,k)=0;
            elseif memAM{arr1}(j+1,k-1)==0 || memAM{arr1}(j+1,k)==0 ||
memAM{arr1}(j+1,k+1)==0
               xoKids{i}(j,k)=0;
            end
         elseif rand<=prob1d && memAM{arr2}(j,k)==1 && (sumCell>=7 || sumCell<=5)
            xoKids{i}(j,k)=1;
         end
      end
   end
   index1=index1+1;
end

%% Mutation operator

nPop=length(xoKids);      %Sets population length same xo population
fPopM=xoKids;             %Sets initial population array same as so population

index2=1;
```

```
for i=1:nPop
   for j=2:matrixs(1)+1
      for k=2:matrixs(2)+1
         %Sets probability values for adding or deleting cell
         rand2=rand(1);
         prob2=0.3;
         prob2d=0.4;
         %Sum cells around current cell
         sumCell=fPopM{i}(j-1,k-1)+fPopM{i}(j,k-1)+fPopM{i}(j+1,k-1)+fPopM{i}(j-1,k)...
            +fPopM{i}(j+1,k)+fPopM{i}(j-1,k+1)+fPopM{i}(j,k+1)+fPopM{i}(j+1,k+1);
         %Checks if mutation can occur, then mutates. checks to make
         %sure a point can be generated or deleted. Generate requires a
         %cell around the current cell to exist. Delete is based on the
         %sum of cells around the current
         if rand2<=prob2 && fPopM{i}(j,k)==1
            if fPopM{i}(j-1,k-1)==0 || fPopM{i}(j-1,k)==0 || fPopM{i}(j-1,k+1)==0
               fPopM{i}(j,k)=0;
            elseif fPopM{i}(j,k-1)==0 || fPopM{i}(j,k+1)==0
               fPopM{i}(j,k)=0;
            elseif fPopM{i}(j+1,k-1)==0 || fPopM{i}(j+1,k)==0 || fPopM{i}(j+1,k+1)==0
               fPopM{i}(j,k)=0;
            end
         elseif rand2<=prob2 && fPopM{i}(j,k)==0 && (sumCell>=7 || sumCell<=5)
            fPopM{i}(j,k)=1;
         end
      end
   end
end

%% Filter Results
n=1;

%Filter goes through results an checks for components that have low
%surrounding cells. If a cell has only 1 or 0 cells around it, it is
%deleted. While this does lose some pertinent information, it is in
%exchange for eliminating most impossible results
for i=1:nPop
  for j=2:matrixs(1)+1
     for k=2:matrixs(2)+1
        sumCell=fPopM{i}(j-1,k-1)+fPopM{i}(j,k-1)+fPopM{i}(j+1,k-1)+fPopM{i}(j-1,k)...
           +fPopM{i}(j+1,k)+fPopM{i}(j-1,k+1)+fPopM{i}(j,k+1)+fPopM{i}(j+1,k+1);
        if fPopM{i}(j,k)==0 && sumCell>=7
           fPopM{i}(j,k)=1;
        end
     end
  end
```

```
end

%% Create final population
%Combines results for final population. 1 is passed through, 2-5 are the
%results from the crossover, mutation and filter

for i=1:5
   if i==1
      fPopF{i}=memAM{sSum(1,2)};
   else
      fPopF{i}=fPopM{i-1};
   end
end

%% Create new generation memory
%Creates baseline file for new generation information. Sets the array
%location information.
for i=1:5
  if i==1
     ngLocs{i}=num2str(memLoc(sSum(1,2)));
     ngLoc(i)=str2num(ngLocs{i});
  else
     ngLocs{i}=[num2str(i),'.',num2str(genNum)];
     ngLoc(i)=str2num(ngLocs{i});
  end
end
ngLoc=ngLoc';

%Writes csv for baseline information
csvwrite(['GenDataG',num2str(genNum),'.csv'],ngLoc);

%% Create BW and Contour Plots

for i=1:5
numstring=num2str(i);

%Sets the names for the figures and curve file
BW=['BWFigure',num2str(i)];
CF=['ContourFigure',num2str(i)];
CvFl=['CurveFile',num2str(i),'.txt'];

%Cuts generation into just the array, ignoring the borders
Array{i}=fPopF{i}(2:11,2:51);
csvwrite(['ShapeArray',numstring,'.csv'],Array{i});

%Plots the binary image and saves
```

```
figure('visible','off')
[r2,c2]=size(fPopF{i});
imagesc((1:c2)+0.5,(1:r2)+0.5,fPopF{i})
colormap(gray);
axis off
axis equal
whitebg
print(1,'-dpng', BW)

%Plots the contour image and saves
figure('visible','off')
[Cont1,Cont2]=contourf(fPopF{i},2);
colormap(hot)
axis on
axis equal
whitebg
print(2,'-dpng',CF)

close(1)
close(2)

%%
%Utilizes exterior program to pull contour information
[x,y,z]=C2xyz(Cont1);
dW=1;
dH=1;

%Finds the length of the contour groups
iter2=0;
l1=length(x);
for i=1:l1
   if z(i)>0.5
      iter2=iter2+1;
      length_c(i)=length(x{1,i});
   end
   length_cell(i)=length(x{1,i});
end

%Finds the maximum length of the cells
maxl=max(length_c);

%Extracts the locations of the contours from the results
iter3=0;
x_loc=zeros(maxl,iter2);
y_loc=zeros(maxl,iter2);
for i=1:l1
```

```
    if z(i)>0.5
        iter3=iter3+1;
        group(iter3)=iter3;
        length_t(iter3)=length(x{1,i});
        for j=1:length_cell(i)
            x_loc(j,iter3)=(50/50)*((x{1,i}(1,j)*dW)-dW*1.5);
            y_loc(j,iter3)=(10/10)*((y{1,i}(1,j)*dH)-dH*1.5);
            point(j,iter3)=j;
        end
    end
end
z_loc=0;
%%
%Prints information into text file formatted for ANSYS

header1='#Group';
header2='Point';
header3='X-cord';
header4='Y-cord';
header5='Z-cord';

fid2=fopen(CvFl,'w');

fprintf(fid2, [ header1 '\t' header2 '\t' header3 '\t' header4 '\t' header5 '\r\n']);
fprintf(fid2,'#Group 1 \r\n');

for i=1:iter3
    for j=1:length_t(i)-1
        fprintf(fid2, '%1.0f \t%2.0f \t%2.2f \t%2.2f \t%1.f \r\n', group(i), point(j,i), x_loc(j,i),
y_loc(j,i), z_loc);
    end
    fprintf(fid2, '%1.0f \t0 \r\n', group(i));
    if i<iter3
        fprintf(fid2, '#Group %1.0f \r\n', group(i+1));
    end
end

fclose(fid2);

end
```

Appendix D: Generational Optimization

```
clear
clc
close all

%% Set the generation number for naming information
genNum=100;
memLen=25;
%% Read the memory and generational arrays
for i=1:memLen
   memA{i}=csvread(['Shape_array',num2str(i),'.csv']);
   if i<=5
      genA{i}=csvread(['ShapeArray',num2str(i),'.csv']);
   end
end
memA=memA';
genA=genA';

%% Import data for memory and generation, extract to variables
memRes=csvread('ExtMem.csv',0);
memLoc=memRes(:,1);
dR=memRes(:,2);
dP=memRes(:,3);
dS=memRes(:,4);
numM=1:length(memA);
matrixs=size(memA{1});

genRes=csvread('GenData.csv',0);
genLoc=genRes(:,1);
dRG=genRes(:,2);
dPG=genRes(:,3);
dSG=genRes(:,4);
numG=1:length(genA);

%% Add borders to system (memory and generation) arrays
for i=1:memLen
   memAM{i}=zeros([matrixs(1)+2 matrixs(2)+2]);
   for j=2:matrixs(1)+1
      for k=2:matrixs(2)+1
         memAM{i}(j,k)=memA{i}(j-1,k-1);
      end
   end
end

for i=1:5
```

```
    genAM{i}=zeros([matrixs(1)+2 matrixs(2)+2]);
    for j=2:matrixs(1)+1
        for k=2:matrixs(2)+1
            genAM{i}(j,k)=genA{i}(j-1,k-1);
        end
    end
end

%% Rank memory solutions
for i=1:length(dRG)
   wSum(i)=dSG(i);
end
cSum=[wSum;numG]';
sSum=sortrows(cSum);

%% Compare solution against memory
randG1=randi([1 memLen]); %Random number for memory access
randG2=randi([1 memLen]); %Random number for memory access
%randG1=13;            %Can set rand values to set variables for rerunning
%randG2=22;
rcheck=1;
%Loop to ensure two different random numbers are used
while rcheck==1
    if randG2==randG1
       randG2=randi([1 20]);
    else
        rcheck=0;
    end
end

%Compares ranked solution 2 against a random memory entropy value. If
%entropy is lower than memory, replaces memory solution with generation
%solution. If memory is lower, replaces the generation solution. Records
%all information for export and use.
if sSum(2,1)<=dS(randG1)
    dS2=sSum(2,1);
    dP2=dPG(sSum(2,2));
    dR2=dRG(sSum(2,2));
    loc2=genLoc(sSum(2,2));
    arv2=genAM{2};
elseif sSum(2,1)>dS(randG1)
    dS2=dS(randG1);
    dP2=dP(randG1);
    dR2=dR(randG1);
    loc2=memLoc(randG1);
    arv2=memAM{randG1};
```

```
end

%Compares ranked solution 4 against a random memory entropy value.
if sSum(4,1)<=dS(randG2)
   dS4=sSum(4,1);
   dP4=dPG(sSum(4,2));
   dR4=dRG(sSum(4,2));
   loc4=genLoc(sSum(4,2));
   arv4=genAM{4};
elseif sSum(4,1)>dS(randG2)
   dS4=dS(randG2);
   dP4=dP(randG2);
   dR4=dR(randG2);
   rp4=0;
   loc4=memLoc(randG2);
   arv4=memAM{randG2};
end

%% Combine new population.
%First solution is passed through. Solution 2 and 4 are pulled from the
%comparison. 3, 5 are from generation.
for i=1:5
  if i==2
     arrF{i}=arv2;
  elseif i==4;
     arrF{i}=arv4;
  else
     arrF{i}=genAM{sSum(i,2)};
  end
end

%% Crossover operations
nxoKids=4;            %Sets the number of crossovers
xoKids=cell(nxoKids,1); %Establishes the number of cells
index1=1;             %Value used to guide crossover selections

%Crossover for 2 to 5
for i=1:nxoKids              %Lists for offspring 2-5
   arr1=sSum(index1,2);      %Sets the value to be used for parent 1
   arr2=sSum(index1+1,2);    %Sets the value to be used for parent 2
   xoKids{i}=arrF{arr1};     %Sets the initial array to parent 1
   for j=2:matrixs(1)+1      %Moves through grid array
      for k=2:matrixs(2)+1
         %Sum cells around current cell of parent 1
         sumCell=arrF{arr1}(j-1,k-1)+arrF{arr1}(j,k-1)+arrF{arr1}(j+1,k-1)+arrF{arr1}(j-1,k)...
            +arrF{arr1}(j+1,k)+arrF{arr1}(j-1,k+1)+arrF{arr1}(j,k+1)+arrF{arr1}(j+1,k+1);
```

```
        %Sets probability values for adding or deleting cell
        rand1=rand(1);
        prob1=0.5;
        prob1d=0.65;

        %Checks if to take parent 2 value, checks to make sure a point
        %can be generated or deleted in parent 1. Generate requires a
        %cell around the current cell to exist. Delete is based on the
        %sum of cells around the current
        if rand1<=prob1 && arrF{arr2}(j,k)==0
           if arrF{arr1}(j-1,k-1)==0 || arrF{arr1}(j-1,k)==0 || arrF{i}(j-1,k+1)==0
              xoKids{i}(j,k)=0;
           elseif arrF{arr1}(j,k-1)==0 || arrF{arr1}(j,k+1)==0
              xoKids{i}(j,k)=0;
           elseif arrF{arr1}(j+1,k-1)==0 || arrF{arr1}(j+1,k)==0 || arrF{arr1}(j+1,k+1)==0
              xoKids{i}(j,k)=0;
           end
        elseif rand<=prob1d && arrF{arr2}(j,k)==1 && (sumCell>=7 || sumCell<=5)
           xoKids{i}(j,k)=1;
        end
     end
  end
  index1=index1+1;
end

%% Mutation operations

nPop=length(xoKids);    %Sets population length same xo population
fPopM=xoKids;           %Sets initial population array same as so population

for i=1:nPop
  for j=2:matrixs(1)+1
     for k=2:matrixs(2)+1
        %Sets probability values for adding or deleting cell
        rand2=rand(1);
        prob2=0.3;
        prob2d=0.4;
        %Sum cells around current cell
        sumCell=fPopM{i}(j-1,k-1)+fPopM{i}(j,k-1)+fPopM{i}(j+1,k-1)+fPopM{i}(j-1,k)...
           +fPopM{i}(j+1,k)+fPopM{i}(j-1,k+1)+fPopM{i}(j,k+1)+fPopM{i}(j+1,k+1);

        %Checks if mutation can occur, then mutates. checks to make
        %sure a point can be generated or deleted. Generate requires a
        %cell around the current cell to exist. Delete is based on the
        %sum of cells around the current
        if rand2<=prob2 && fPopM{i}(j,k)==1
```

```
            if fPopM{i}(j-1,k-1)==0 || fPopM{i}(j-1,k)==0 || fPopM{i}(j-1,k+1)==0
               fPopM{i}(j,k)=0;
            elseif fPopM{i}(j,k-1)==0 || fPopM{i}(j,k+1)==0
               fPopM{i}(j,k)=0;
            elseif fPopM{i}(j+1,k-1)==0 || fPopM{i}(j+1,k)==0 || fPopM{i}(j+1,k+1)==0
               fPopM{i}(j,k)=0;
            end
         elseif rand2<=prob2 && fPopM{i}(j,k)==0 && (sumCell>=7 || sumCell<=5)
            fPopM{i}(j,k)=1;
         end
      end
   end
end

%% Filter Results
n=1;

%Filter goes through results an checks for components that have low
%surrounding cells. If a cell has only 1 or 0 cells around it, it is
%deleted. While this does lose some pertinent information, it is in
%exchange for eliminating most impossible results
for i=1:nPop
  for j=2:matrixs(1)+1
     for k=2:matrixs(2)+1
        sumCell=fPopM{i}(j-1,k-1)+fPopM{i}(j,k-1)+fPopM{i}(j+1,k-1)+fPopM{i}(j-1,k)...
            +fPopM{i}(j+1,k)+fPopM{i}(j-1,k+1)+fPopM{i}(j,k+1)+fPopM{i}(j+1,k+1);
        if fPopM{i}(j,k)==0 && sumCell>=7
           fPopM{i}(j,k)=1;
        end
     end
  end
end

%% Create final population
%Combines results for final population. 1 is passed through, 2-5 are the
%results from the crossover, mutation and filter

for i=1:5
   if i==1
      fPopF{i}=genAM{sSum(1,2)};
   else
      fPopF{i}=fPopM{i-1};
   end
end

%% Create new memory file
```

```
%Coalesces data to create new memory information. Replaces or keeps any
%data compared against the randomly selected solution. All else remains
%constant
for i=1:memLen
   if i==randG1
      dSEM(i)=dS2;
      dPEM(i)=dP2;
      dREM(i)=dR2;
      locEM(i)=loc2;
   elseif i==randG2
      dSEM(i)=dS4;
      dPEM(i)=dP4;
      dREM(i)=dR4;
      locEM(i)=loc4;
   else
      dSEM(i)=dS(i);
      dPEM(i)=dP(i);
      dREM(i)=dR(i);
      locEM(i)=memLoc(i);
   end
end

%Writes memory to csv file
EMWrite=[locEM', dREM', dPEM', dSEM'];
csvwrite(['ExtMemGen',num2str(genNum),'.csv'],EMWrite);

%% Create new generation memory
%Creates baseline file for new generation information. Sets the array
%location information.
for i=1:5
  if i==1
     ngLocs{i}=num2str(genLoc(sSum(1,2)));
     ngLoc(i)=str2num(ngLocs{i});
  else
     ngLocs{i}=[num2str(i),'.',num2str(genNum)];
     ngLoc(i)=str2num(ngLocs{i});
  end
end
ngLoc=ngLoc';

%Writes csv for baseline information
csvwrite(['GenDataG',num2str(genNum),'.csv'],ngLoc);

%Writes csv to provide values for memory comparison for this generation
randNum=[randG1, randG2];
csvwrite(['RandNum.csv'],randNum);
```

```
%% Create BW and Contour Plots
for i=1:5
numstring=num2str(i);

%Sets the names for the figures and curve file
BW=['BWFigure',num2str(i)];
CF=['ContourFigure',num2str(i)];
CvFl=['CurveFile',num2str(i),'.txt'];

%Cuts generation into just the array, ignoring the borders
Array{i}=fPopF{i}(2:11,2:51);
csvwrite(['ShapeArray',numstring,'.csv'],Array{i});

%Plots the binary image and saves
figure('visible','off')
[r2,c2]=size(fPopF{i});
imagesc((1:c2)+0.5,(1:r2)+0.5,fPopF{i})
colormap(gray);
axis off
axis equal
whitebg
print(1,'-dpng', BW)

%Plots the contour image and saves
figure('visible','off')
[Cont1,Cont2]=contourf(fPopF{i},2);
colormap(hot)
axis on
axis equal
whitebg
print(2,'-dpng',CF)

close(1)
close(2)

%%
%Utilizes exterior program to pull contour information
[x,y,z]=C2xyz(Cont1);
dW=1;
dH=1;

%Finds the length of the contour groups
iter2=0;
l1=length(x);
for i=1:l1
```

```
    if z(i)>0.5
        iter2=iter2+1;
        length_c(i)=length(x{1,i});
    end
    length_cell(i)=length(x{1,i});
end

%Finds the maximum length of the cells
maxl=max(length_c);

%Extracts the locations of the contours from the results
iter3=0;
x_loc=zeros(maxl,iter2);
y_loc=zeros(maxl,iter2);
for i=1:l1
    if z(i)>0.5
        iter3=iter3+1;
        group(iter3)=iter3;
        length_t(iter3)=length(x{1,i});
        for j=1:length_cell(i)
            x_loc(j,iter3)=(50/50)*((x{1,i}(1,j)*dW)-dW*1.5);
            y_loc(j,iter3)=(10/10)*((y{1,i}(1,j)*dH)-dH*1.5);
            point(j,iter3)=j;
        end
    end
end
z_loc=0;
%%
%Prints information into text file formatted for ANSYS

header1='#Group';
header2='Point';
header3='X-cord';
header4='Y-cord';
header5='Z-cord';

fid2=fopen(CvFl,'w');

fprintf(fid2, [ header1 '\t' header2 '\t' header3 '\t' header4 '\t' header5 '\r\n']);
fprintf(fid2,'#Group 1 \r\n');

for i=1:iter3
    for j=1:length_t(i)-1
        fprintf(fid2, '%1.0f \t%2.0f \t%2.2f \t%2.2f \t%1.f \r\n', group(i), point(j,i), x_loc(j,i),
y_loc(j,i), z_loc);
    end
```

```
    fprintf(fid2, '%1.0f \t0 \r\n', group(i));
    if i<iter3
        fprintf(fid2, '#Group %1.0f \r\n', group(i+1));
    end
end

fclose(fid2);

end
```