

Marshall University

## Marshall Digital Scholar

---

Theses, Dissertations and Capstones

---

2019

# An Inference-Driven Branch and Bound Optimization Strategy for Planning Ambulance Services

Kevin McDaniel

mcdaniel55@marshall.edu

Follow this and additional works at: <https://mds.marshall.edu/etd>



Part of the [Emergency and Disaster Management Commons](#), [Mathematics Commons](#), and the [Non-linear Dynamics Commons](#)

---

### Recommended Citation

McDaniel, Kevin, "An Inference-Driven Branch and Bound Optimization Strategy for Planning Ambulance Services" (2019). *Theses, Dissertations and Capstones*. 1258.

<https://mds.marshall.edu/etd/1258>

This Thesis is brought to you for free and open access by Marshall Digital Scholar. It has been accepted for inclusion in Theses, Dissertations and Capstones by an authorized administrator of Marshall Digital Scholar. For more information, please contact [zhangj@marshall.edu](mailto:zhangj@marshall.edu), [beachgr@marshall.edu](mailto:beachgr@marshall.edu).

**AN INFERENCE-DRIVEN BRANCH AND BOUND OPTIMIZATION  
STRATEGY FOR PLANNING AMBULANCE SERVICES**

A thesis submitted to  
the Graduate College of  
Marshall University  
In partial fulfillment of  
the requirements for the degree of  
Master of Arts

in

Mathematics

by

Kevin McDaniel

Approved by

Dr. Michael Schroeder, Committee Chairperson

Dr. Raid Al-Aqtash

Dr. Scott Sarra

Marshall University  
December 2019

## APPROVAL OF THESIS/DISSERTATION

We, the faculty supervising the work of Kevin Curtis McDaniel, affirm that the thesis, *An Inference-Driven Branch And Bound Optimization Strategy For Planning Ambulance Services*, meets the high academic standards for original scholarship and creative work established by the Department of Mathematics and the College of Science. This work also conforms to the editorial standards of our discipline and the Graduate College of Marshall University. With our signatures, we approve the manuscript for publication.



Dr. Michael Schroeder, Department of Mathematics Committee Chairperson

12/17/19

Date



Dr. Raid Al-Aqtash, Department of Mathematics Committee Member

12/17/19

Date



Dr. Scott Sarra, Department of Mathematics Committee Member

12/17/19

Date

## TABLE OF CONTENTS

List of Figures .....	vi
List of Tables .....	vii
Abstract .....	viii
Chapter 1    Ambulance Planning and Optimization .....	1
Chapter 2    Topics from Optimization .....	2
2.1    Multi-Objective Optimization .....	2
2.1.1    Pareto Dominance and Pareto Optimality .....	3
2.1.2    Aggregative Scoring .....	3
2.2    Branch and Bound .....	6
2.2.1    The Structure of the Algorithm .....	7
2.2.2    Example: Minimization with Ideal Bounding .....	8
Chapter 3    Results from Probability and Statistics .....	11
3.1    The Left-Continuous Inverse .....	11
3.2    Sampling and Error Distributions .....	11
3.3    Linear Combinations of Independent Normal Distributions .....	13
3.4    Extreme Value Theory .....	13
3.4.1    Extreme Value Distributions and Domains of Attraction .....	13
3.4.2    Fisher-Tippett-Gnedenko Theorem .....	14
3.4.3    Pickands-Balkema-De Haan Theorem .....	15
3.5    Empirical Distributions .....	16
3.6    Bootstrap Estimation .....	16
3.6.1    Example: Error Distribution of the Sample Mean .....	18
Chapter 4    Methods from Numerical Integration .....	19
4.1    Newton-Cotes Rules .....	20
4.1.1    Trapezoid Rule .....	21
4.1.2    Simpson's 3/8 Rule .....	22



4.2	Gaussian Quadrature .....	23
4.2.1	Gauss-Legendre Quadrature .....	23
4.2.2	Gauss-Hermite Quadrature .....	25
4.3	Clenshaw-Curtis Quadrature .....	27
Chapter 5	Overview of Inference-Driven Branch and Bound .....	30
5.1	The Bounding Criterion .....	31
5.2	Guiding Parameters and Statistics .....	31
5.2.1	Arithmetic Mean .....	32
5.2.2	Small Quantiles .....	33
5.2.3	Sub-Quantile Means .....	33
5.2.4	$1/n^{th}$ Quantile and the Sample Minimum .....	34
5.3	Inputs and Outputs .....	34
Chapter 6	Comparing Branches .....	36
6.1	Overview .....	36
6.2	Comparison by Monte Carlo Estimation of Proportion .....	37
6.3	Pairwise Comparisons .....	38
6.3.1	Guiding Statistics with Normally Distributed Errors .....	38
6.4	Comparisons Involving More Than Two Branches .....	42
6.4.1	Derivation of the Discard Probability Integral .....	43
6.4.2	Alternative Forms for The Discard Probability Integral .....	44
6.4.3	Fast Computation of the Discard Probability Integral .....	45
6.4.4	Determining the Interval of Integration .....	46
6.5	Comparison of Quadrature Methods for Computing Discard Probabilities .....	47
6.6	Discarding with a Confidence Level .....	49
Chapter 7	Model Fitting Method for Extreme Quantile Guiding Parameters .....	52
7.1	On Bootstrapping the Error Distribution of the Sample Minimum .....	52
7.2	Estimating Upper Tail Moments .....	53
7.3	Estimating the Tail Distribution .....	56
7.4	Estimating the Parameter Distribution .....	58

Chapter 8	Test Functions .....	60
8.1	Revisiting Ideal Bounding .....	60
8.2	The Wicked Comb .....	60
8.3	The Eggholder Test Function .....	64
Chapter 9	Application to Ambulance Planning .....	66
9.1	Data Used .....	67
9.1.1	Wayne 911 Data .....	67
9.1.2	Open Street Map (OSM) Data .....	68
9.2	Goals and Metrics .....	68
9.3	EMS Simulation Model .....	70
9.3.1	Assumptions and Simplifications .....	71
9.3.2	Implementation .....	72
9.3.3	Generated Data .....	74
9.4	Optimization .....	75
9.4.1	Branching .....	75
9.5	Results .....	77
9.6	Future Work .....	77
References	.....	81
Appendix A	Letter from Institutional Research Board .....	83

## LIST OF FIGURES

Figure 1	Progression Diagram of Branch and Bound Optimization.....	10
Figure 2	The ECDF of a Sample .....	17
Figure 3	Comparison of CLT and Bootstrap Error Estimates .....	18
Figure 4	Nodes and Weights for a 7-point Trapezoid Rule .....	21
Figure 5	Nodes and Weights for a 7-point Simpson's 3/8 Rule .....	22
Figure 6	Nodes and Weights for a 7-point Gauss-Legendre Rule .....	24
Figure 7	Nodes and Weights for a 7-point Gauss-Hermite Rule .....	25
Figure 8	Nodes and Weights for a 7-point Clenshaw-Curtis Quadrature Rule .....	28
Figure 9	Example of the Sets $L$ and $U$ . .....	42
Figure 10	Convergence Rate Comparison of Quadrature Methods .....	50
Figure 11	Graph of $f(n) = \left(\frac{n-1}{n}\right)^n$ for $n$ up to 20 .....	53
Figure 12	Graph of the Modified Excess ECDF for Fitting an Upper-Tail Estimate ...	55
Figure 13	Example of an ECDF with a Tail Estimate .....	58
Figure 14	Progression Diagram for the Optimization from Chapter 2 .....	61
Figure 15	Graph of the Wicked Comb Test Function .....	62
Figure 16	Progression Diagram for the Wicked Comb .....	63
Figure 17	Perspective View of the Eggholder Test Function .....	64
Figure 18	Progression Diagram for the Eggholder .....	65
Figure 19	Flowchart of Simulated Ambulance Behavior .....	74
Figure 20	Heatmap of the Solution Space for Plans with 10 Ambulances .....	76
Figure 21	Progression Diagram for an Application of Branch and Bound .....	78

## LIST OF TABLES

Table 1	Example of Aggregative Scoring in Comparing Foods for Purchase .....	4
Table 2	Computations for Gauss-Legendre Quadrature .....	24
Table 3	Computations for Gauss-Hermite Quadrature .....	27
Table 4	Computations for Clenshaw-Curtis Quadrature .....	29
Table 5	Comparison of Relative Errors for 7-point Quadrature Rules .....	29

## **ABSTRACT**

Strategic placement of ambulances is important to the efficient functioning of emergency services. As part of an ongoing collaboration with Wayne County 911, we developed a strategy to optimize the placement of ambulances throughout Wayne County based on de-identified call and response data from 2016 and 2017. The primary goals of the optimization were minimizing annual operating cost and mean response time, as well as providing a constructive solution that could naturally evolve from the existing plan. This thesis details the derivation and implementation of one of the optimization strategies used in this project. It is based on parametric statistical inference and the “Branch and Bound” pattern that is often used in non-linear global optimization.

## CHAPTER 1

### AMBULANCE PLANNING AND OPTIMIZATION

In the spring of 2016, faculty at Marshall University were approached by Wayne County 911 to collaborate on finding ways to improve the planning of their ambulance services, following the recommendations of the EMS white paper [18]. Over the course of 2016 and 2017, this developed into a project aimed at determining an efficient plan that describes where ambulances should be located throughout the county, and during what times they should be active, so that the cost and response time are simultaneously minimized. There are many possible approaches for this optimization, and in many cases the optimal plans may not always exhibit a clear progression. For instance, it is possible that an optimal solution for planning the distribution of 3 ambulances has no placements in common with an optimal solution for planning the distribution of 5 ambulances. For this reason, in this thesis we develop an algorithm that constructs an optimal plan for  $k$  ambulances by placing ambulances one-by-one until the desired number of ambulances is reached. Such a model can be used to inform the growth of the existing ambulance system toward an optimal plan with more ambulances. It also sheds light on the relative importance of each ambulance placement, which can be used to inform the design of contingency plans in the event that an ambulance must be removed from service.

In this thesis, we describe factors related to solving this planning problem, motivate and develop a technique for finding well-optimized solutions with a natural progression, and briefly discuss the results of this method. In Chapters 2, 3, and 4, we discuss a variety of mathematical topics that will be important in the development of our optimization method. Minor simplifications have been made throughout for the sake of brevity. In Chapters 5, 6, and 7, we construct a routine for solving single-objective derivative-free optimization problems with mild assumptions on the structure of the solution space. In Chapters 8 and 9, we apply that routine to several optimization problems, including the ambulance planning problem, and analyze the results.

## CHAPTER 2

### TOPICS FROM OPTIMIZATION

#### 2.1 Multi-Objective Optimization

**Multi-objective optimization** is the process of finding acceptable solutions to an optimization problem when those solutions must be evaluated against more than one goal, or objective. It is conventional to refer to an optimization problem as multi-objective only when the goals conflict; that is, when the partial ordering of solutions that is induced by one goal sometimes disagrees with the partial ordering induced by another goal. A handy example is given by Bader [1], in the form of purchasing a device to be used as a word processor. Objectives in that problem might include both to maximize the comfort with which one can type on the device, and to maximize the mobility of the device. Ideally, achievement of these two goals would be correlated, as in the case when comparing an antique typewriter to a modern laptop, where the laptop is both more portable and more comfortable to type on. Commonly, however, these goals are in conflict. A desktop PC may be easier to type on but less portable than a laptop, which is likewise easier to type on but less portable than a smart phone. As this example demonstrates, it may be the case that no solution will be simultaneously optimal with respect to every objective, so the process of selecting a single “best” solution necessarily involves deciding on appropriate tradeoffs between the objectives, which is something that can be difficult to define mathematically. When making that decision, it is informative to have a sense of what the best possible outcomes are for each objective and for each combination of objectives. With this in mind, the goal of a multi-objective optimization is to find a representative set of solutions that describe the best possible outcomes with respect to all conceivable weightings of the objectives - usually referred to as the solution set. Ideally, a solution set should satisfy three minimal requirements. First, the solution set should not be too large, so that a decision maker can readily consider the solutions together without being overwhelmed. Secondly, the solution set should be diverse, so that a sense of the available alternatives can be obtained from observing only it. Lastly, the solution set should be a subset of the Pareto-optimal set, as defined in the proceeding subsection. In our brief description of multi-objective optimization, we will focus on two of the

most accessible and relevant topics: Pareto optimality and aggregative scoring.

### 2.1.1 Pareto Dominance and Pareto Optimality

Let  $A$  and  $B$  be solutions of an  $n$ -objective optimization in a **solution space**  $\Omega$ , which is the set of all feasible solutions. **Pareto dominance** is a relation on the solution space that describes when one solution is clearly preferable to another. Denote the event that  $A$  is preferred to  $B$  in the  $i^{\text{th}}$  objective by  $A_i \prec B_i$ . It is said that  $A$  Pareto-dominates  $B$ , written  $A \prec_{\text{par}} B$ , if and only if  $A_i \prec B_i$  for some  $i = 1, 2, \dots, n$  and  $B_j \not\prec A_j$  for all  $j = 1, 2, \dots, n$ . In words, a solution  $A$  Pareto-dominates a solution  $B$  if  $A$  is better than  $B$  in some way, and  $B$  is not better than  $A$  in any way. If one had to decide only between these two solutions,  $A$  would be a good choice no matter which objectives are considered, because  $A$  is never worse than  $B$  in any respect. Furthermore, in a comparison where the  $i^{\text{th}}$  objective is given positive weight,  $A$  will be unambiguously preferable to  $B$ . The Pareto dominance relation induces a strict partial ordering on  $\Omega$ .

A related characteristic called **Pareto optimality** is used to identify solutions that are potentially optimal before any tradeoffs between objectives are specified. A solution  $A \in \Omega$  is called **Pareto optimal**, or **Pareto efficient**, if for every other solution  $B \in \Omega$  such that  $A$  is worse than  $B$  in at least one objective,  $A$  is also better than  $B$  in at least one objective. If a solution  $B$  is chosen over the Pareto optimal solution  $A$ , it can be a tradeoff or a downgrade, but can never be an overall upgrade - any improvement in one objective will come at the cost of inferior performance in another objective. Pareto optimality is also characterized in terms of Pareto dominance: A solution  $A$  is Pareto optimal if and only if there is no other solution  $B \in \Omega$  such that  $B \prec_{\text{par}} A$ . In allusion to this characterization, it is common to refer to Pareto optimal solutions as “non-dominated” solutions. The set of all Pareto optimal solutions in the solution space is called the **Pareto set**.

### 2.1.2 Aggregative Scoring

One of the simplest methods for making multi-objective optimization problems more soluble is to decide on a set of tradeoff functions that relate the objectives. The result is a reduction of the problem to an optimization in fewer objectives - ideally just one - at the cost of



Item	Weight (lb)	Price (\$)	Energy Content (kCal)	Enjoyment (1 to 5)	Nutrition (1 to 5)
Beef Shank	2	12	1160	4	3
Bag of Candy	0.5	2.5	850	5	1
Whole Watermelon	4	2	470	3	2
Box of Pasta Noodles	1	0.8	1600	2	2
Bunch of Kale	0.5	2.5	113	2	5

**Table 1: Example of Aggregative Scoring in Comparing Foods for Purchase**

Each food is listed with its estimated scores for each objective. Every item in the list is Pareto-Optimal under our set of goals.

an arbitrary narrowing of the Pareto-optimal set. This is generally done by creating **objective functions**, which each combine some set of measurements describing the quality of a solution with respect to the objectives into a single number. Such methods are referred to as **aggregative scoring**, as they aggregate quality metrics for several objectives into one score. This reduction simplifies the problem, but it also reduces the level of detail with which we view relationships between solutions. If a solution  $A$  is better than a solution  $B$  in one way, but worse in another, then they are incomparable under the Pareto dominance relation. However, aggregative scoring will assign each of them to a single score, which may prescribe that  $A$  is better or worse than  $B$ . The following example illustrates how this affects the relative optimality of solutions.

When shopping for groceries, we must balance the need to minimize price, minimize energy content, maximize enjoyment, and maximize nutritional benefit. Table 1 lists a variety of foods that are available, as well as their prices and estimates of their energy content, enjoyability, and nutritional benefit.

One way to score these is to prioritize getting the greatest immediate enjoyment per dollar. Under this scoring method, we have

$$f(\text{food}) = \frac{\text{cost}}{\text{enjoyment}}. \quad (2.1)$$

This induces the ordering

$$\text{Noodles} \prec \text{Candy} \prec \text{Watermelon} \prec \text{Kale} \prec \text{Beef}.$$

Somewhat counter-intuitively, despite prioritizing enjoyment, we prefer noodles to candy and kale to beef because cost is equally a priority. If the score is further weighted by the amount (in pounds) of the item purchased, a different ordering is obtained:

$$f(\text{food}) = \frac{\text{cost}}{\text{weight} \times \text{enjoyment}}. \quad (2.2)$$

The associated ordering is

$$\text{Watermelon} \prec \text{Noodles} \prec \text{Candy} \prec \text{Beef} \prec \text{Kale}.$$

This scoring method assigns lower preferences to more nutritious foods. A shopper who values enjoyment equally with a healthy diet might instead use

$$f(\text{food}) = \frac{\text{cost} \times \text{energy}}{\text{weight} \times (\text{enjoyment} + \text{nutrition})}. \quad (2.3)$$

We assume for this example that the units of preference given for enjoyment and nutrition are of similar magnitude. This yields the ordering

$$\text{Watermelon} \prec \text{Kale} \prec \text{Noodles} \prec \text{Candy} \prec \text{Beef}.$$

Each of these aggregative scoring methods makes a clear indication about which foods are more optimal than others, even though they are incomparable by Pareto dominance. Each scoring method induces a total ordering on the solution space, and this ordering is often inconsistent between scoring methods. Not every Pareto optimal solution will be optimal under an aggregative scoring method. However, any solution that is optimal under an aggregative scoring method must be Pareto optimal. As a result, the set of optimal solutions under an aggregative scoring method can be viewed as a narrowing of the Pareto set, and can be obtained by finding the subset of the Pareto set that is preferred under the scoring method. Aggregative scoring methods are simplifications through which we can more easily make sense of the complex relationships between solutions. The utility of an aggregative scoring method is determined by how well the total

ordering it induces on  $\Omega$  reflects our preferences in the comparisons we want to perform.

## 2.2 Branch and Bound

Suppose we are interested in solving a global optimization problem with one objective. In general, this means we want to minimize an objective function  $f : \Omega \rightarrow \mathbb{R}$  operating on elements of the solution space  $\Omega$ . If  $|\Omega|$  is finite and small, and evaluating  $f$  is acceptably fast, the minimum can be computed with absolute certainty by performing an exhaustive search. If, instead,  $|\Omega|$  is very large, and we may evaluate  $f$  for only a small fraction of the elements of  $\Omega$  in the available time, we must decide which elements we evaluate and which others we ignore.

**Branch and Bound** is a pattern in algorithm design that provides a reasonable way to decide which elements to ignore. It was originally proposed by Land and Doig in 1960 [13].

Appropriately enough, it is composed of the repeated application of two processes, which are called **branching** and **bounding**.

The branching process splits a solution set  $S$  into a collection of proper subsets,  $\text{Branch}(S) = \{S_1, S_2, \dots, S_n\}$ , that forms a cover of  $S$ . Each of these subsets is called a **branch**. Ideally, this can be recursed on the subsets as well, so that  $S_1$  can be branched into  $S_{1,1}, S_{1,2}, \dots, S_{1,n}$  and likewise for the other branches of  $S$ . The number of times this recursion must be able to be performed is problem-specific. The function  $\text{Branch}(S)$  need not be defined for every conceivable set  $S$ ; it only needs to be defined for the sets to which it will be applied. Branching by partition, so that the branches are disjoint, is often desirable so that the differences between the branches are clearer. It can be convenient to think of  $\text{Branch}(S)$  as forming a partition of  $S$ , but the output generally only needs to be a cover of  $S$ .

The bounding process determines whether or not each branch should be kept or discarded. It can be defined as a function  $\text{Bound}(\mathcal{S})$  that maps a set of branches  $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$  to a subset of itself. Each branch in  $\text{Bound}(\mathcal{S})$  is kept, while each branch in  $\mathcal{S} \setminus \text{Bound}(\mathcal{S})$  is “discarded”, which means it will not be branched further. Bounding is often performed by estimating bounds on the properties of the optimal solution, and using that to decide whether or not a branch is likely to contain elements that satisfy those bounds. If it is determined that a branch probably does not contain optimal elements, then that branch is discarded. Some of its

elements may still be eligible to be evaluated, however, if they also belong to another branch that was not discarded. This approach of culling less promising branches is used in Chapter 3.

Alternatively, if only one optimal solution is desired, and the bounding process can determine with confidence that one or more branches contain optimal elements, then all other branches may be discarded so that the search space can be reduced to those branches only. This is the approach used in bisection search [21].

In the slightly simplified case where branching yields a partition, the branching process can be thought of as creating a tree from subsets of the set of feasible solutions, with every solution belonging to the root node and where for a set of solutions  $\mathcal{A}$ ,  $\text{Branch}(\mathcal{A})$  generates a new set of nodes that are linked to  $\mathcal{A}$ . The efficiency of the branch and bound pattern arises from the judicious pruning of this tree, so that only the most worthwhile branches are examined in detail.

The specifics of both branching and bounding are determined by the problem; there is often more than one feasible method for branching, and many possible bounding criteria. The choice of branching pattern for a set of solutions may be influenced, for example, by the desire to construct a solution in a particular step-by-step way. Another goal may be to avoid overlapping elements between the branches, so that the process can be visualized as generating a tree. Bounding criteria may be informed by prior knowledge about the problem, or come entirely from ad-hoc heuristics.

Additional details on branch and bound optimization can be found in Mehlhorn and Sanders [17].

### 2.2.1 The Structure of the Algorithm

The basic structure of branch and bound optimization is given by the following instructions:

**Step 1: Initialization.**

Set  $\mathcal{S} = \{\Omega\}$ , so that  $\mathcal{S}$  is a collection containing the set of all feasible solutions,  $\Omega$ .

**Step 2: Branching.**

Set  $\mathcal{S} = \bigcup_{S \in \mathcal{S}} \text{Branch}(S)$ , so that  $\mathcal{S}$  is now the set of all the branches of its previous elements.

**Step 3: Bounding.**

Set  $\mathcal{S} = \text{Bound}(\mathcal{S})$ . This generally involves evaluating the solution sets in  $\mathcal{S}$ , comparing them in some manner, and then removing any branches that are determined to be unfavorable from the search process.

**Step 4: Condition Checking.**

If termination conditions are met, the result is  $\mathcal{S}$ . Otherwise, go to step 2 (Branching).

General termination conditions may include

1. Being unable to continue to branch. If the  $\text{Branch}()$  function is undefined for some of the elements of  $\mathcal{S}$ , then the algorithm cannot proceed.
2. Having too many solution sets in  $\mathcal{S}$  to evaluate all of their branches in the available time during the next bound step. If the bounding step requires a significant amount of computation time, and it must be performed for a very large number of branches, then completion of the routine may become unfeasible.
3. Few enough solutions remain in  $\bigcup_{S \in \mathcal{S}} S$  that they can be evaluated exhaustively in a reasonable timespan. If an exhaustive search of the remaining solutions can be performed, it may be more reliable than the bounding function at locating optimal solutions.

Each of these conditions indicates that it may be beneficial to continue the optimization with an alternative strategy. Additional termination conditions may depend on the specific problem being solved.

**2.2.2 Example: Minimization with Ideal Bounding**

In this example, we demonstrate the overall process of branch and bound optimization. Suppose we have the real-valued function  $f(x) = 1 - \cos\left(\frac{\pi x^2}{2}\right)$  defined on the closed interval  $[0, 3.2]$ , and we want to use a branch and bound strategy to locate the set  $M$  of inputs that minimize  $f$ ,

$$M = \{s \in [a, b] \mid f(s) \leq f(x) \ \forall x \in [a, b]\}. \quad (2.4)$$

Suppose we have a bounding function  $\text{Bound}(\mathcal{S})$  that discards a solution set  $S$  if and only if it does not contain a minimum:<sup>1</sup>

$$\text{Bound}(\mathcal{S}) = \{S \in \mathcal{S} \mid M \cap S \neq \emptyset\}. \quad (2.5)$$

We define  $\text{Branch}(S)$  for a closed interval  $S = [a, b]$  to be

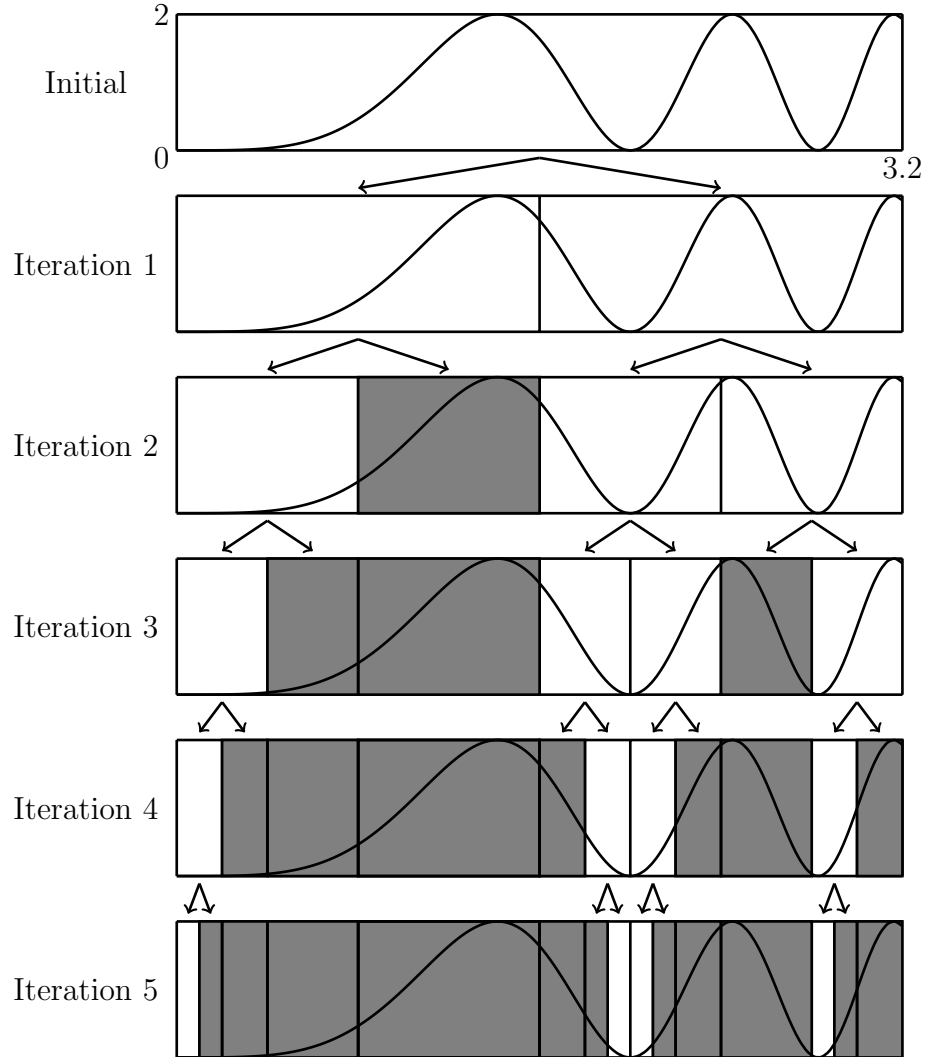
$$\text{Branch}(S) = \left\{ \left[ a, \frac{a+b}{2} \right], \left[ \frac{a+b}{2}, b \right] \right\}. \quad (2.6)$$

Note that  $\text{Branch}()$  is only defined here for closed intervals. This is sufficient, since every time  $\text{Branch}()$  is used, the input is a closed interval. The solution space  $\Omega \equiv [0, 3.2]$  is a closed interval, and  $\text{Branch}()$  always produces a set of closed intervals, so the first and every subsequent evaluation of  $\text{Branch}()$  will be performed on a closed interval.

In the first two iterations of branch and bound, the solution space  $[0, 3.2]$  is branched into  $[0, 1.6]$ ,  $[1.6, 3.2]$ , and both intervals are kept during bounding because  $[0, 1.6]$  contains a minimum at 0 and  $[1.6, 3.2]$  contains minima at 2 and  $2\sqrt{2}$ . Each of these intervals is then branched to yield  $\mathcal{S} = \{[0, 0.8], [0.8, 1.6], [1.6, 2.4], [2.4, 3.2]\}$ . The interval  $[0.8, 1.6]$  does not contain a minimum, so it is discarded, leaving  $\mathcal{S} = \{[0, 0.8], [1.6, 2.4], [2.4, 3.2]\}$  at the end of the second iteration. For the purpose of demonstration, let the termination condition be that five bounding steps have been completed. The state of the process after each iteration is summarized by Figure 1. After the third iteration, each additional iteration reduces the size of the remaining search space by half. If the midpoint of the two bounds of each remaining interval is taken to be an estimate of the location of a minimum, then these estimates will converge geometrically to the complete set of minima.

---

<sup>1</sup>While it is convenient, this definition may not always be practical, as determining whether or not a solution set contains a minimum may be very difficult depending on the choices of  $f$ ,  $a$ , and  $b$ . We label this an “ideal” bounding function to highlight that simplification.



**Figure 1: Progression Diagram of Branch and Bound Optimization**

This is a visualization of branch and bound to minimize  $f(x)$  over  $[0, 3.2]$ , with minima at 0, 2, and  $2\sqrt{2}$ . Discarded sets are greyed out, and an arrow is drawn from each retained set  $S$  to each element of  $\text{Branch}(S)$  in the next iteration.

## CHAPTER 3

### RESULTS FROM PROBABILITY AND STATISTICS

#### 3.1 The Left-Continuous Inverse

For any nondecreasing function  $f$ , define the **left-continuous inverse** [7] of  $f$  to be

$$f^{\leftarrow}(x) := \inf\{y : f(y) \geq x\}. \quad (3.1)$$

This is a convenient notation for inverting a cumulative distribution function (CDF), denoted by  $F(x)$ , into a quantile function  $F^{\leftarrow}(x)$  when the associated probability density function (PDF)  $f$  has finite support. In agreement with the usual notation, this yields  $F^{-1}(x)$  when  $f$  has unbounded support.

#### 3.2 Sampling and Error Distributions

Let  $S$  be a sample of size  $n$  from a random variable  $X$ . Suppose we want to estimate the value of a parameter  $\theta$  of  $X$  using an estimating statistic  $\hat{\theta}_n$  computed from the  $S$ . Depending on the particular observations in  $S$ , the resulting value of  $\hat{\theta}_n$  may vary. The probability distribution of the value of  $\hat{\theta}_n$  is called the **sampling distribution** of  $\hat{\theta}_n$  [24]. Perhaps the most intuitive characterization of the sampling distribution of  $\hat{\theta}_n$  is as the asymptotic distribution of  $\hat{\theta}_n$  under repeated observation. That is, if observations of  $\hat{\theta}_n$  are made repeatedly by taking additional samples of size  $n$  from  $X$  and computing  $\hat{\theta}_n$  from a new sample each time, the sampling distribution of  $\hat{\theta}_n$  is the distribution of all values of  $\hat{\theta}_n$  as the number of samples goes to infinity.

The signed error of an estimate  $\hat{\theta}$  of a parameter  $\theta$  is given by the difference

$$\hat{\theta} - \theta. \quad (3.2)$$

Under repeated observation with a fixed sample size  $n$ , the estimator  $\hat{\theta}_n$  is a random variable that follows its sampling distribution and the difference (3.2) takes on a probability distribution called



the **error distribution** of  $\hat{\theta}_n$ , denoted here by the random variable  $e_n$ ,

$$e_n = \hat{\theta}_n - \theta. \quad (3.3)$$

Several common statistics have normal sampling and error distributions for large  $n$ , and of particular importance in this work are the sampling distributions of the mean and the non-extremal quantiles of a population.

### Sampling Distribution of the Mean

Let  $P$  be a population with mean  $\mu$  and finite variance  $\sigma^2$ . An estimate of the mean of  $P$  from a sample  $X_1, X_2, \dots, X_n$  is given by the sample mean  $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$ , and the sampling distribution of  $\bar{X}$  is given by

$$\bar{X} \sim \mathcal{N}(\mu, \sigma^2/n) \quad (3.4)$$

with error distribution

$$\bar{X} - \mu \sim \mathcal{N}(0, \sigma^2/n) \quad (3.5)$$

for sufficiently large  $n$ . Exactly how large  $n$  must be to qualify as “sufficiently large” depends on the distribution of the population. Wackerly et al. suggest a practical threshold of  $n \geq 30$  [24].

This result is often called the **Central Limit Theorem** (CLT).

### Sampling Distributions of Quantiles

If a continuous random variable  $X$  has density  $f(x)$  and CDF  $F(x)$ , then in the estimation of the  $p$ th quantile  $\mathfrak{Q}_p := F_X^{-1}(p)$ , where  $p \in (0, 1)$ , the sampling distribution of the sample quantile  $\hat{\mathfrak{Q}}_p$  for sufficiently large  $n$  is given by

$$\hat{\mathfrak{Q}}_p \sim \mathcal{N}\left(\mathfrak{Q}_p, \frac{p(1-p)}{nf(\mathfrak{Q}_p)^2}\right) \quad (3.6)$$

with error distribution

$$\hat{\mathfrak{Q}}_p - \mathfrak{Q}_p \sim \mathcal{N}\left(0, \frac{p(1-p)}{nf(\mathfrak{Q}_p)^2}\right). \quad (3.7)$$

Note that the sampling distributions of the extreme quantiles 0 and 1 are excluded from

this definition. Indeed, error distributions for extreme quantiles tend not to be normally distributed, as we discuss in Section 3.4. Since accurate estimation of the density  $f$  from sample data is difficult unless additional information about the distribution is known ahead of time, it may be more practical to obtain an estimate of the variance of this normal distribution via bootstrap. It should be noted that “sufficiently large” is more and more difficult to achieve as  $p$  gets close to 0 or 1. As such, while theoretically there is always a sample size beyond which (3.6) holds, it may be impractically large for quantiles near the extremes. A proof of this result is given by De Haan and Ferreira [7].

### 3.3 Linear Combinations of Independent Normal Distributions

The following result appears as Theorem 6.3 in Wackerly [24]. Let  $X_1, X_2, \dots, X_n$  be independent, normally distributed random variables with means  $E(X_i) = \mu_i$  and variances  $V(X_i) = \sigma_i^2$  for  $i = 1, 2, \dots, n$ , and let  $a_1, a_2, \dots, a_n$  be constants. If

$$U = \sum_{i=1}^n a_i X_i = a_1 X_1 + a_2 X_2 + \dots + a_n X_n, \quad (3.8)$$

then  $U$  is a normally distributed random variable with

$$E(U) = \sum_{i=1}^n a_i \mu_i = a_1 \mu_1 + a_2 \mu_2 + \dots + a_n \mu_n \quad (3.9)$$

and

$$V(U) = \sum_{i=1}^n a_i^2 \sigma_i^2 = a_1^2 \sigma_1^2 + a_2^2 \sigma_2^2 + \dots + a_n^2 \sigma_n^2. \quad (3.10)$$

Effectively, this means that the set of normally distributed random variables is closed under addition and scalar multiplication.

### 3.4 Extreme Value Theory

#### 3.4.1 Extreme Value Distributions and Domains of Attraction

The following definition is from De Haan and Ferreira [7], with minor reformatting. Let  $X_1, X_2, \dots, X_n$  be independent, identically distributed random variables with a common

underlying distribution function  $F$ , and define  $x^* := \sup\{x : F(x) < 1\}$ . Then

$$P(\max(X_1, X_2, \dots, X_n) \leq x) = P(X_1 \leq x, X_2 \leq x, \dots, X_n \leq x) = F^n(x),$$

so  $\max(X_1, X_2, \dots, X_n)$  converges in probability to  $x^*$  as  $n \rightarrow \infty$ . If there exists a sequence of constants  $a_n > 0$  and  $b_n \in \mathbb{R}$ , for  $n = 1, 2, \dots$  such that

$$\frac{\max(X_1, X_2, \dots, X_n) - b_n}{a_n}$$

has a nondegenerate probability distribution as  $n \rightarrow \infty$ ; that is,

$$\lim_{n \rightarrow \infty} F^n(a_n x + b_n) = G(x) \quad (3.11)$$

for every continuity point  $x$  of  $G$ , where  $G$  is a nondegenerate distribution function. Then  $G(x)$  is an **extreme value distribution**.

In somewhat more practical terms: Let  $G_n(x)$  be the distribution of the maximum of a sample of size  $n$  from a distribution  $X$ . If, after disregarding location and scale, the shape of  $G_n(x)$  converges to  $G(x)$  as  $n \rightarrow \infty$ , then  $G(x)$  is an extreme value distribution. An enormous body of literature is dedicated to showing that the shapes of the distributions of maxima and minima for samples from nearly all distributions converge to extreme value distributions for sufficiently large samples [12]. Such distributions  $X$  for which  $G_n(x) \rightarrow G(x)$  as  $n \rightarrow \infty$  are said to be in the **domain of attraction** of  $G$ , written  $X \in \mathcal{D}(G)$ .

### 3.4.2 Fisher-Tippett-Gnedenko Theorem

An extremely valuable theoretical result was obtained first by Fisher and Tippet (1928), then later generalized by Gnedenko (1943) [12]. It fully characterizes the class of extreme value distributions as  $G_\gamma(ax + b)$ , where  $a > 0$  and  $b \in \mathbb{R}$ , and

$$G_\gamma(x) = \exp\left(-(1 + \gamma x)^{-1/\gamma}\right), \quad 1 + \gamma x > 0. \quad (3.12)$$

This means that if  $X \in \mathcal{D}(G)$  (that is, the shape of the distributions of sample maxima

converges for large  $n$ ), then  $G$  must be  $G_\gamma$  for some real  $\gamma$  under an appropriate shift and rescaling. These distributions  $G_\gamma$  are called **generalized extreme value** (GEV) distributions. The GEV distributions generalize the Weibull, Fréchet, and Gumbel distributions, with each corresponding to different choices of  $\gamma$ . See De Haan and Ferreira for details [7].

### 3.4.3 Pickands-Balkema-De Haan Theorem

Another powerful theorem from Extreme Value Theory was formalized in 1974 by Balkema and De Haan [2]. It states that if  $X \in \mathcal{D}(G)$  - the same mild assumption as Fisher-Tippett-Gnedenko - the upper tail of  $F_X$  can be approximated using a **generalized Pareto distribution** (GPD). The CDF of a GPD with location  $\mu$ , shape parameter  $\xi$ , and scale  $\sigma > 0$  is given by

$$F_{GPD}(x; \mu, \sigma, \xi) = \begin{cases} 1 - \left(1 + \xi \left(\frac{x-\mu}{\sigma}\right)\right)^{-1/\xi} & \text{if } \xi \neq 0, \text{ and} \\ 1 - e^{-(x-\mu)/\sigma} & \text{if } \xi = 0. \end{cases} \quad (3.13)$$

If  $X_1, X_2, \dots, X_n$  are i.i.d. random variables with common distribution  $X$  and continuous CDF  $F(x)$ , and  $X \in \mathcal{D}(G)$  for some GEV distribution  $G$ , then the conditional excess distribution function,

$$F_u(x) = P(X - u \leq x \mid X > u) = \frac{F(u+x) - F(u)}{1 - F(u)}, \quad (3.14)$$

is well-approximated by the CDF of a GPD with location  $\mu = 0$  for suitably large  $u$  [12].

The parameter  $u$  can be interpreted as the transition point such that  $F(x)$  is well-approximated by a transformed GPD CDF for  $x \geq u$ . The difficulty in applying this model arises from the need to find optimal choices for  $u$ ,  $\xi$ , and  $\sigma$  so that the model fits the data as closely as possible. This is nontrivial, as each of these parameters strongly influences the optimality of the others. In the following year, Pickands [10] developed an efficient computational method for estimating these optimal values in  $O(n^2)$  time. This allows us to model the tail of a distribution with a GPD approximation using observed values of  $X_1, X_2, \dots, X_n$ . Our implementation builds on the approach outlined by Pickands. See Chapter 7 for details.

### 3.5 Empirical Distributions

The empirical distribution of a univariate dataset  $D = \{x_1, x_2, \dots, x_n\}$ , indexed so that  $x_i \leq x_{i+1}$  for  $i < n$ , is a probability distribution with its CDF described by the piecewise step function [7] given below:

$$\hat{F}_{Emp}(x) = \begin{cases} 0 & \text{if } x < x_1, \\ i/n & \text{if } x_i \leq x < x_{i+1}, \text{ for } i = 1, 2, \dots, n-1, \text{ and} \\ 1 & \text{if } x \geq x_n. \end{cases} \quad (3.15)$$

This CDF is called the **empirical CDF** of the data, often abbreviated as the ECDF. If  $X_1, X_2, \dots, X_n$  are i.i.d. following a common distribution  $X$ , then the ECDF of observations  $x_1, x_2, \dots, x_n$  will converge pointwise to  $X$  as  $n \rightarrow \infty$ . As such, if a large sample is taken from a population, the ECDF of the sample can be used as an approximation of the CDF of the population.

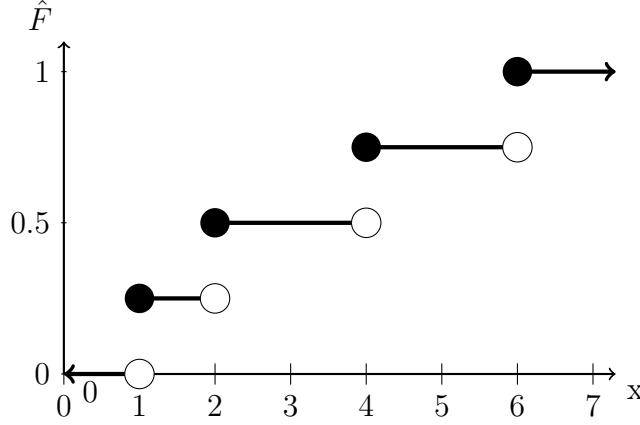
The ECDF can be characterized as a step function that starts at 0 coming from the left, increases by  $1/n$  for each datapoint it travels over, and then continues on to the right at 1. Its corresponding density function is the superposition of  $n$  Dirac delta functions that have each been rescaled by a factor of  $1/n$ . That is,

$$\hat{f}_{Emp}(x) = \frac{1}{n} \sum_{i=1}^n \delta(x_i). \quad (3.16)$$

The ECDF of a small dataset is demonstrated in Figure 2.

### 3.6 Bootstrap Estimation

Bootstrap estimation is a Monte Carlo method for estimating the distributions of the parameters of a probability distribution using limited sample data. The pattern of logic behind bootstrap is that since the empirical distribution of the sample can be treated as an approximation of the distribution of the population, properties of a sample of size  $n$  taken from the population can be modeled by observing the properties of samples of size  $n$  taken from the empirical distribution. As it may be expensive to observe new values from the population, but



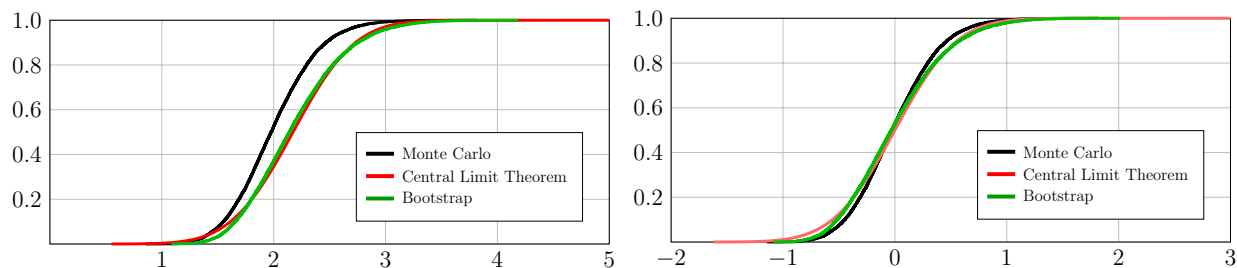
**Figure 2: The ECDF of a Sample**

This is the ECDF of a sample given by the set  $\{1, 2, 4, 6\}$ .

sampling from already observed data takes little effort, a few hundred of these “bootstrap” samples can be used to model the shape and spread of sampling distributions with a very modest amount of computational effort. A practical guide to bootstrap is provided by Press et. al. [20], and much of this information can be found in greater detail there.

Let  $S_0 = X_1, X_2, \dots, X_n$  be a sample from a probability distribution  $X$  with parameter  $\theta$ . Suppose  $\hat{\theta}$  is an estimator of  $\theta$ . Computing  $\hat{\theta}$  from  $S_0$ , we have one value of  $\hat{\theta}$ , with an unknown error of estimation  $\hat{\theta} - \theta$ . For  $j = 1, 2, \dots, m$  define **bootstrap samples**  $S_j$  by drawing a sample of  $n$  elements with replacement from  $S_0$ . Note that the bootstrap samples  $S_1, S_2, \dots, S_m$  may differ from  $S_0$  if  $n > 1$ . Let  $\hat{\theta}_i$  be the value of  $\hat{\theta}$  computed from  $S_i$ . If the empirical distribution of  $S_0$  is a good approximation of  $X$  for the purpose of estimating  $\theta$ , then the distribution of  $\hat{\theta} - \theta$  approximately follows that of  $\hat{\theta}_j - \hat{\theta}_0$  for  $j = 1, 2, \dots, m$ . Since we can choose  $m$  to be as large as we want, we can cheaply sample from the approximate error distribution  $\hat{\theta}_j - \hat{\theta}_0$  to develop an image of how  $\hat{\theta} - \theta$  behaves.

The usefulness of this technique is dependent on the quality of the ECDF of  $S_0$  as an approximation of the CDF of  $X$  in the computation of  $\hat{\theta}$ . For this reason, larger original sample sizes (values of  $n$ ) and statistics that are computed from a large number of values spread out over the support tend to yield better bootstrap estimates. Conversely, the error distributions of statistics computed from singular or closely-clustered points tend to be harder to bootstrap. After a point, taking additional bootstrap samples (increasing  $m$ ) yields little improvement in the



**Figure 3: Comparison of CLT and Bootstrap Error Estimates**

(Left) A comparison of the CDFs of two estimates of the sampling distribution of the mean, as well as the actual sampling distribution (via Monte Carlo). (Right) The CDFs of the corresponding error distributions, centered at 0.

accuracy of parameter estimates from the approximate error distribution  $\hat{\theta}_j - \hat{\theta}_0$ . It is conventional to choose  $m \approx 200$  for most computations [9].

### 3.6.1 Example: Error Distribution of the Sample Mean

As we mentioned in Section 3.2 the error in the sample mean  $\bar{x}$  for sufficiently large  $n$  is given by the Central Limit Theorem (CLT) (3.5) as

$$\bar{x} - \mu \sim \mathcal{N}(0, \sigma^2/n).$$

So we should expect the variance of the estimator to be close to  $\sigma^2/n$ . Figure 3 details the exact (via Monte Carlo), CLT, and bootstrap estimates of the sampling distribution of the mean of an exponential distribution with  $\lambda = 0.5$  with  $n = 30$ . The CLT and bootstrap models are centered at the sample mean, which in this example is somewhat to the right of the true mean at 2. This difference of location does not affect the estimated error distributions, which are good models of the true error distribution even with our small sample size and the highly skewed shape of the exponential distribution.

## CHAPTER 4

### METHODS FROM NUMERICAL INTEGRATION

**Numerical integration** is the computational approximation of the value of a definite integral. It is sometimes alternatively called **quadrature**. Much of the information in this section is given in greater detail by Sauer [21]. In general, we want to evaluate

$$I(f) = \int_a^b f(x) \, dx \tag{4.1}$$

for some function  $f$  that is integrable on a nondegenerate interval  $[a, b]$ . For the purposes of this work, we will focus on the case where this interval of integration is closed and bounded, but many of these methods can also be used on open or unbounded intervals. In much of the literature on this subject, the interval is taken to be  $[-1, 1]$ , and an appropriate transformation from  $[a, b]$  to  $[-1, 1]$  is used to define a new function  $g(x)$  on  $[-1, 1]$  such that

$$I(f) = \int_{-1}^1 g(x) \, dx. \tag{4.2}$$

Such a transformation can always be found, even if the integral is improper, so the interval of integration may be chosen to be  $[-1, 1]$  without loss of generality [22]. A simple way to do this for proper integrals is to define a function  $h : [-1, 1] \rightarrow [a, b]$ ,

$$h(x) = \frac{1}{2} ((b - a)x + b + a) \tag{4.3}$$

so that  $g = f \circ h$  linearly maps  $[-1, 1]$  to  $f([a, b])$ . For improper integrals, things are somewhat trickier. It is sometimes possible to truncate the interval of integration by discarding one or more infinite subintervals. If this cannot be done, then Gauss-Laguerre or Gauss-Hermite quadrature (for single and double-sided improper integrals, respectively) may work, but some argue that Gaussian integration methods are not reliable for evaluating improper integrals [22]. Definite integrals on nondegenerate intervals can be thought of as taking uncountably many points into consideration, but computational methods can only involve finitely many steps, so it is necessary



to determine a finite set of inputs  $x_1, x_2, \dots, x_n$  at which the integrand should be evaluated. Each function value  $g(x_i)$  may then be scaled by an appropriate weight factor  $w_i$ , and the result is summed to approximate the definite integral as follows:

$$I(f) \approx \sum_{i=1}^n w_i g(x_i) \equiv I_n(f). \quad (4.4)$$

Intuitively, evaluating the function at a greater number of points will tend to yield a better approximation of the integral.

This much is common to all of the quadrature techniques we will consider. The computational differences between these techniques arise mainly from choosing different evaluation points  $x_i$  and weights  $w_i$  for  $i = 1, 2, \dots, n$ .

It is worth noting that the theoretical efficiency of a quadrature technique is commonly judged by the highest degree of polynomial with arbitrary coefficients that it can integrate exactly using  $n$  evaluation points [21]. While interesting from an analytical standpoint, this metric can be misleading [22] and was disregarded for this project.

In Chapter 6, we develop a specific set of integrals on which we may apply these methods. For the purpose of demonstration, in this chapter we provide an example of each technique for the definite integral

$$\int_1^4 \sqrt{x} \, dx, \quad (4.5)$$

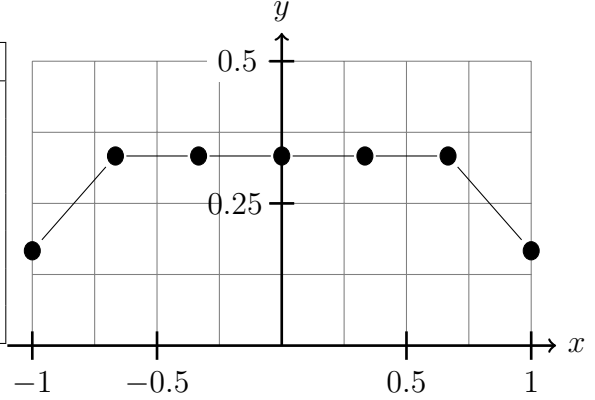
which evaluates to  $4\sqrt{6}$ . Each example will use the same number of evaluation points so that a comparison may be more clearly drawn between the methods.

## 4.1 Newton-Cotes Rules

Newton-Cotes quadrature techniques are obtained by interpolating a function's values at evenly spaced points with polynomials, and then analytically integrating the interpolant over the interval of interest. They are closely related to Riemann integration. The evaluation positions  $x_1, x_2, \dots, x_n$  are defined to be evenly spaced on  $[a, b]$ , with  $x_1$  taking the value of the left bound of the interval and  $x_n$  likewise taking the value of the right bound. Though others were also examined for this project, two particularly effective Newton-Cotes rules are described in this

**Evaluation Points for the Trapezoid Rule**

Nodes $x_i$	Weight $w_i$
-1	0.166 666 666 666 667
-0.666 666 666 666 667	0.333 333 333 333 333
-0.333 333 333 333 333	0.333 333 333 333 333
0	0.333 333 333 333 333
0.333 333 333 333 333	0.333 333 333 333 333
0.666 666 666 666 667	0.333 333 333 333 333
1	0.166 666 666 666 667



**Figure 4: Nodes and Weights for a 7-point Trapezoid Rule**

The interval of integration is assumed to be  $[-1, 1]$  to allow comparison with other methods.

section.

#### 4.1.1 Trapezoid Rule

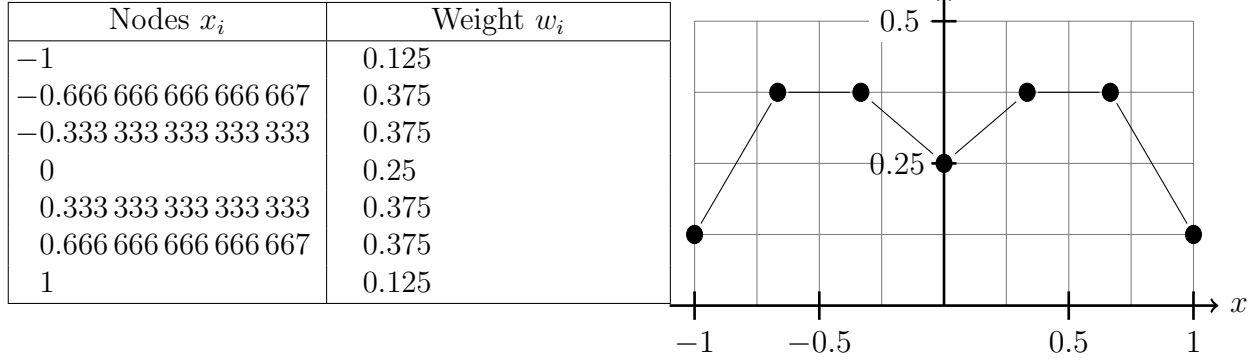
The first Newton-Cotes rule we consider is the trapezoid rule - a relatively simple numerical integration technique that is often mentioned in elementary calculus. In the trapezoid rule, each adjacent pair of evaluation points is interpolated linearly to construct a trapezoidal region under the interpolant. The value of the definite integral is the area under the curve defined by the integrand, and this is approximated geometrically by summing the trapezoids' areas. The weights are given by  $w_1 = w_n = h/2$ , and  $w_2 = w_3 = \dots = w_{n-1} = h$  where  $h = (b - a)/(n - 1)$ . An example of the nodes and weights for a 7-point trapezoid rule is given in Figure 4. Despite being relatively slow to converge for most expressions, this technique is famously efficient for integrating periodic functions [23].

Applying this to the example integral (4.5) with 7 evaluation points, we have

$$I_7(x) = \sum_{i=1}^7 w_i \sqrt{x_i} = h \left[ \frac{1}{2} \sqrt{1} + \sqrt{1.5} + \sqrt{2} + \sqrt{2.5} + \sqrt{3} + \sqrt{3.5} + \frac{1}{2} \sqrt{4} \right] \approx \frac{1}{2} (9.32298) = 4.66149.$$

The error in this approximation is  $5.178 \times 10^{-3}$ , and the relative error is  $1.11 \times 10^{-3}$ .

**Evaluation Points for Simpson's 3/8 Rule**



**Figure 5: Nodes and Weights for a 7-point Simpson's 3/8 Rule**

The interval of integration is assumed to be  $[-1, 1]$  to allow comparison with other methods.

#### 4.1.2 Simpson's 3/8 Rule

Simpson's 3/8 rule is another Newton-Cotes method, and it is usually much faster than the trapezoid rule with similar reliability. It is obtained by interpolating sequences of four consecutive points by cubic polynomials, and the number of evaluation points  $n$  must satisfy  $n = 3k + 1$  for some positive integer  $k$ . There are a few different variations of this rule, and the one we implemented is defined by the weights in the following expression

$$w_i = \begin{cases} \frac{3h}{8} & \text{if } i = 1 \text{ or } i = n, \\ \frac{6h}{8} & \text{if } 1 < i < n \text{ and } i \bmod 3 = 1, \text{ and} \\ \frac{9h}{8} & \text{otherwise,} \end{cases} \quad (4.6)$$

where  $h$  is the fixed step size  $(b - a)/(n - 1)$ .

Applying the nodes and weights in Figure 5 to the integral in (4.5) using a transformation from  $[-1, 1]$  to  $[1, 4]$ , a Simpson's 3/8 Rule approximation of (4.5) using 7 evaluation points would

be given by

$$\begin{aligned}
\int_1^4 \sqrt{x} \, dx &\approx \sum_{i=1}^7 w_i \sqrt{x_i} \\
&= \frac{3h}{8} \left[ 1\sqrt{1} + 3\sqrt{1.5} + 3\sqrt{2} + 2\sqrt{2.5} + 3\sqrt{3} + 3\sqrt{3.5} + 1\sqrt{4} \right] \\
&\approx \frac{3}{8} \left( \frac{1}{2} \right) (24.8878) \\
&= 4.66646.
\end{aligned}$$

The error in this approximation is  $2.058 \times 10^{-4}$ , yielding a relative error of  $4.41 \times 10^{-5}$ .

## 4.2 Gaussian Quadrature

Gaussian quadrature is based on evaluating the transformed function  $g(x)$  at the roots of orthogonal polynomials. For proper integrals, the Legendre polynomials are often used, while for improper integrals the Laguerre and Hermite polynomials are standard [6].

### 4.2.1 Gauss-Legendre Quadrature

For Gauss-Legendre quadrature, the integral  $I(f)$  must be transformed so that the interval of integration is over  $[-1, 1]$ , as in (4.4).

The evaluation points  $x_1, x_2, \dots, x_n$  of an  $n$ -point Gauss-Legendre rule are given by the roots of the  $n$ th-degree Legendre polynomial, while the weights  $w_i$  are computed from that polynomial's derivative. The 0th and 1st-degree Legendre Polynomials are  $P_0 = 1$  and  $P_1 = x$ , respectively, and higher degree Legendre polynomials can be obtained from these by the recurrence relation

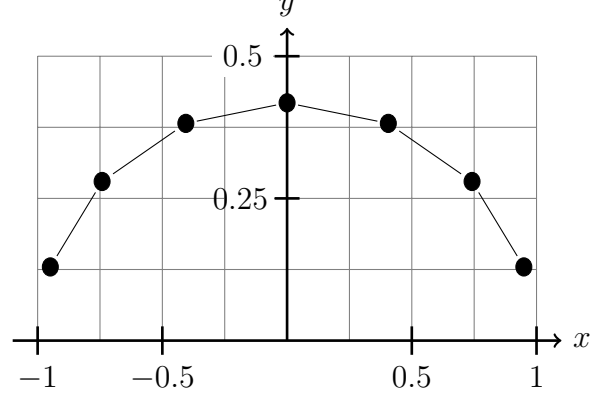
$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x). \quad (4.7)$$

An efficient method for computing the nodes and weights was developed by Bogaert [4]. Our implementation of Gauss-Legendre quadrature uses an efficient wrapper of the C++ code that was published alongside that paper to obtain the necessary nodes and weights.

For our example integral, first the standard interval of integration  $[-1, 1]$  is transformed to

**Evaluation Points for Gauss Legendre**

Node $x_i$	Weight $w_i$
-0.949 107 912 342 758	0.129 484 966 168 870
-0.741 531 185 599 394	0.279 705 391 489 277
-0.405 845 151 377 397	0.381 830 050 505 119
0	0.417 959 183 673 469
0.405 845 151 377 397	0.381 830 050 505 119
0.741 531 185 599 394	0.279 705 391 489 277
0.949 107 912 342 758	0.129 484 966 168 870



**Figure 6: Nodes and Weights for a 7-point Gauss-Legendre Rule**

**Computations for Gauss Legendre on (4.8)**

Node $x_i$	Weight $w_i$	$h(x_i)$	$\sqrt{h(x_i)}$	$w_i\sqrt{h(x_i)}$
-0.949 ...	0.129 ...	1.076 338 131 485 86	1.037 467 171 281 03	0.134 336 401 574 637
-0.741 ...	0.279 ...	1.387 703 221 600 91	1.178 008 158 545 99	0.329 495 233 163 667
-0.405 ...	0.381 ...	1.891 232 272 933 90	1.375 220 808 791 78	0.525 100 630 876 654
0	0.417 ...	2.5	1.581 138 830 084 19	0.660 851 494 696 412
0.405 ...	0.381 ...	3.108 767 727 066 10	1.763 169 795 302 23	0.673 231 211 989 349
0.741 ...	0.279 ...	3.612 296 778 399 09	1.900 604 319 262 45	0.531 609 275 185 514
0.949 ...	0.129 ...	3.923 661 868 514 14	1.980 823 532 905 98	0.256 486 868 144 832

**Table 2: Computations for Gauss-Legendre Quadrature**

These are the computations for applying a 7-point Gauss-Legendre rule to the example integral (4.5).

$[1, 4]$  by the linear mapping  $h(x) = 1.5x + 2.5$ . Then our integral (4.5) can be rewritten

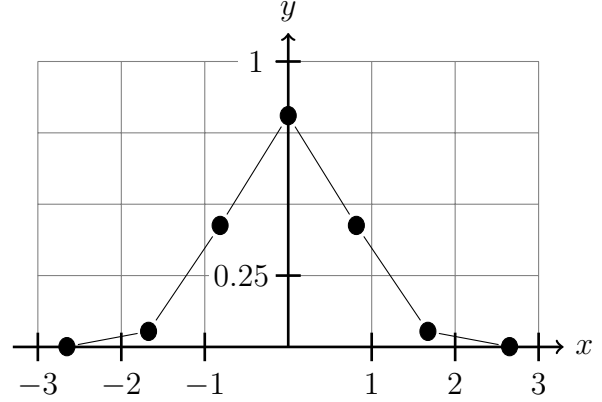
$$\int_1^4 \sqrt{x} \, dx = \frac{dh}{dx} \int_{-1}^1 \sqrt{h(x)} \, dx \approx \frac{3}{2} \sum_{i=1}^n w_i \sqrt{h(x_i)}. \quad (4.8)$$

The nodes and weights for a 7-point Gauss-Legendre rule are given in Figure 6, and the computations for (4.8) are demonstrated in Table 2. Applying these computations, we have

$$\sum_{i=1}^7 w_i \sqrt{h(x_i)} = 3.11111111563107,$$

**Evaluation Points for Gauss Hermite**

Node $x_i$	Weight $w_i$
-2.651 961 356 835 23	0.000 971 781 245 099
-1.673 551 628 767 47	0.054 515 582 819 127
-0.816 287 882 858 965	0.425 607 252 610 128
0	0.810 264 617 556 808
0.816 287 882 858 965	0.425 607 252 610 128
1.673 551 628 767 47	0.054 515 582 819 127
2.651 961 356 835 23	0.000 971 781 245 099



**Figure 7: Nodes and Weights for a 7-point Gauss-Hermite Rule**

and by multiplying by the derivative  $\frac{dh}{dx}$ , our approximation of the definite integral (4.5) becomes

$$I_n = \frac{dh}{dx} \sum_{i=1}^7 w_i \sqrt{h(x_i)} = 4.6666666734466.$$

This yields an error of  $6.7799 \times 10^{-9}$  and a percent error of  $1.452 \times 10^{-9}$ .

#### 4.2.2 Gauss-Hermite Quadrature

Gauss-Hermite quadrature computes an approximation of an integral over  $\mathbb{R}$  of the form

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^n w_i f(z_i). \quad (4.9)$$

The nodes of an  $n$ th-order Gauss-Hermite quadrature rule are located at the roots of the physicist's Hermite polynomial with degree  $n$ . A reasonably efficient method for computing the nodes and weights was implemented in C++ by J. Burkardt [5], and this implementation was ported to C# for this project [16]. The nodes and weights for a 7-point rule are demonstrated by Figure 7.

We examine this technique because it appears, at a glance, to be particularly well-suited to evaluating integrals of the form encountered in Chapter 6. However, it is not well-suited to

evaluating proper integrals. Nonetheless, we define the transformation

$$x : (-\infty, \infty) \rightarrow [1, 4], \quad x(z) = \frac{3}{\pi} \tan^{-1} \left( \frac{\pi}{3} z \right) + 2.5 \quad (4.10)$$

and observe that

$$\frac{dx}{dz} = \frac{3}{\pi} \cdot \frac{1}{1 + \frac{(\pi z)^2}{9}} \cdot \frac{\pi}{3} = \frac{9}{9 + (\pi z)^2}. \quad (4.11)$$

Then by a change of variables, we have that

$$\int_1^4 \sqrt{x} \, dx = \int_{-\infty}^{\infty} \sqrt{x(z)} \frac{dx}{dz} dz = \int_{-\infty}^{\infty} \sqrt{x(z)} \frac{9}{9 + (\pi z)^2} dz. \quad (4.12)$$

Since this lacks the required exponential term, we rewrite it as

$$\int_1^4 \sqrt{x} \, dx = \int_{-\infty}^{\infty} e^{-z^2} e^{z^2} \sqrt{x(z)} \frac{9}{9 + (\pi z)^2} dz. \quad (4.13)$$

Then the function  $f$  in (4.9) is

$$f(z) = e^{z^2} \sqrt{\frac{3}{\pi} \tan^{-1} \left( \frac{\pi}{3} z \right) + 2.5} \frac{9}{9 + (\pi z)^2}. \quad (4.14)$$

Approximations using this integrand with a large number of evaluation points do converge to the correct value of the integral with sufficiently precise arithmetic, but not in double-precision, where the  $\exp(z^2)$  term rapidly inflates the size of the subtotal to where rounding errors become significant. It is possible there are alternative transformations  $x(z)$  which enable faster convergence, though Gauss-Hermite is generally a poor choice for evaluating proper integrals like (4.5). In contrast, this technique excels when the integrand is a polynomial in  $z$  multiplied by  $e^{z^2}$ , and is a natural choice for improper integrals where the integrand already contains  $e^{-z^2}$  as a factor. For this reason, a more appropriate example is

$$\int_{-\infty}^{\infty} e^{-x^2} \sqrt{|x|} \, dx, \quad (4.15)$$

**Computations for Gauss-Hermite on (4.8)**

Node $x_i$	Weight $w_i$	$f(x_i)$	$w_i f(x_i)$
-4.101 337 596 178 64	$4.825\,731\,850\,073 \times 10^{-8}$	2.025 175 942 030 38	$9.772\,956\,045\,457 \times 10^{-8}$
-3.246 608 978 372 41	$2.043\,036\,040\,270 \times 10^{-5}$	1.801 834 892 095 39	$3.681\,213\,623\,168 \times 10^{-5}$
-2.519 735 685 678 24	$1.207\,459\,992\,719 \times 10^{-3}$	1.587 367 533 269 54	$1.916\,682\,790\,164 \times 10^{-3}$
-1.853 107 651 601 51	$2.086\,277\,529\,616 \times 10^{-2}$	1.361 288 966 972 67	$2.840\,026\,583\,110 \times 10^{-2}$
-1.220 055 036 590 75	$1.403\,233\,206\,870 \times 10^{-1}$	1.104 561 015 331 77	$1.549\,956\,695\,727 \times 10^{-1}$
-0.605 763 879 171 06	$4.216\,162\,968\,985 \times 10^{-1}$	0.778 308 344 533 87	$3.281\,474\,820\,676 \times 10^{-1}$
0	$6.043\,931\,879\,211 \times 10^{-1}$	0	0
0.605 763 879 171 06	$4.216\,162\,968\,985 \times 10^{-1}$	0.778 308 344 533 87	$3.281\,474\,820\,676 \times 10^{-1}$
1.220 055 036 590 75	$1.403\,233\,206\,870 \times 10^{-1}$	1.104 561 015 331 77	$1.549\,956\,695\,727 \times 10^{-1}$
1.853 107 651 601 51	$2.086\,277\,529\,616 \times 10^{-2}$	1.361 288 966 972 67	$2.840\,026\,583\,110 \times 10^{-2}$
2.519 735 685 678 24	$1.207\,459\,992\,719 \times 10^{-3}$	1.587 367 533 269 54	$1.916\,682\,790\,164 \times 10^{-3}$
3.246 608 978 372 41	$2.043\,036\,040\,270 \times 10^{-5}$	1.801 834 892 095 39	$3.681\,213\,623\,168 \times 10^{-5}$
4.101 337 596 178 64	$4.825\,731\,850\,073 \times 10^{-8}$	2.025 175 942 030 38	$9.772\,956\,045\,457 \times 10^{-8}$

**Table 3: Computations for Gauss-Hermite Quadrature**

These are computations for applying a 13-point Gauss-Hermite rule to (4.15). Due to the integrand being even, this only uses the information of 7 evaluation points, which is comparable to our examples in the other methods.

which evaluates to approximately 1.22541670247. For this integral, the function  $f$  in (4.9) is

$$f(x) = \sqrt{|x|}. \quad (4.16)$$

An approximation of (4.15) can be obtained by Gauss-Hermite quadrature as

$$I(f) \approx \sum_{i=1}^n w_i \sqrt{|x_i|}. \quad (4.17)$$

The relevant computations are given by Table 3. Adding up the last column, we have an approximate value of 1.02699402025491, which yields an error of  $1.984 \times 10^{-1}$  and a relative error of  $1.619 \times 10^{-1}$ .

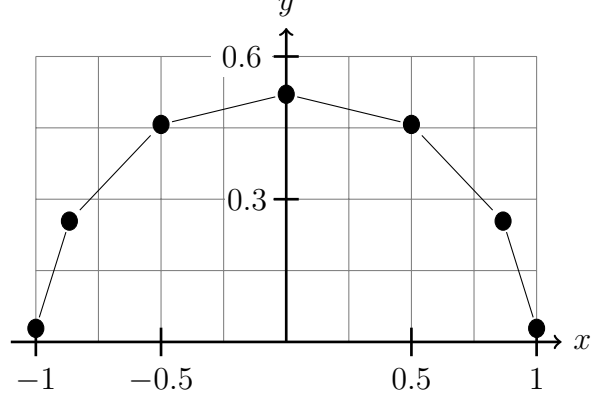
### 4.3 Clenshaw-Curtis Quadrature

Developed in 1960 specifically for use in computer algorithms [6], Clenshaw-Curtis quadrature is very similar to Gauss-Legendre in that it approximates a definite integral with an assumed interval of integration of  $[-1, 1]$ . However, unlike in Gaussian quadrature methods,



**Evaluation Points for Clenshaw Curtis**

Node $x_i$	Weight $w_i$
-1	0.028 571 428 571 428
-0.866 025 403 784 438	0.253 968 253 968 254
-0.5	0.457 142 857 142 857
0	0.520 634 920 634 921
0.5	0.457 142 857 142 857
0.866 025 403 784 438	0.253 968 253 968 254
1	0.028 571 428 571 428



**Figure 8: Nodes and Weights for a 7-point Clenshaw-Curtis Quadrature Rule**

Clenshaw-Curtis nodes have the convenient property of “nesting”, which allows the evaluation points of an  $n + 1$  point rule to be recycled when increasing the number of evaluation points to  $2n + 1$ , allowing the accuracy of the approximation to be refined very efficiently.

Clenshaw-Curtis quadrature rules require evaluating the integrand at Chebyshev points. The nodes  $x_k$  of an  $n + 1$  point rule are given by

$$x_k = \cos\left(\frac{k\pi}{n}\right), \quad k = 0, 1, \dots, n. \quad (4.18)$$

Geometrically, these are the abscissas of  $n + 1$  points evenly spaced along the upper half of a unit circle centered at the origin. A method for efficiently computing the weights of Clenshaw-Curtis rules using the discrete cosine transform was developed by Waldvogel [25], which gives the following computations:

$$w_k = \frac{c_k}{n} \left( 1 - \sum_{j=1}^{n/2} \frac{b_j}{4j^2 - 1} \cos\left(\frac{2jk\pi}{n}\right) \right), \quad k = 0, 1, \dots, n \quad (4.19)$$

where

$$b_j = \begin{cases} 1 & \text{if } j = n/2, \text{ and} \\ 2 & \text{if } j < n/2, \end{cases} \quad \text{and} \quad c_k = \begin{cases} 1 & \text{if } k = 0 \bmod n, \text{ and} \\ 2 & \text{otherwise.} \end{cases} \quad (4.20)$$

Since this method performs integration over the same interval (4.2) as Gauss-Legendre, we will re-use the transformed integrand from that method (4.8) for the example integral. Recall

**Computations for Clenshaw-Curtis on (4.8)**

Node $x_i$	Weight $w_i$	$h(x_i)$	$\sqrt{h(x_i)}$	$w_i\sqrt{h(x_i)}$
-1	0.028 ...	1	1	0.028 571 428 571 428
-0.866 ...	0.253 ...	1.200 961 894 323 34	1.095 884 069 746 13	0.278 319 763 745 049
-0.5	0.457 ...	1.75	1.322 875 655 532 30	0.604 743 156 814 764
0	0.520 ...	2.5	1.581 138 830 084 19	0.823 196 089 313 673
0.5	0.457 ...	3.25	1.802 775 637 731 99	0.824 126 005 820 340
0.866 ...	0.253 ...	3.799 038 105 676 66	1.949 112 132 658 52	0.495 012 605 119 625
1	0.028 ...	4	2	0.057 142 857 142 857

**Table 4: Computations for Clenshaw-Curtis Quadrature**

These are computations for applying a 7-point Clenshaw-Curtis rule to the example integral (4.5).

Method	Relative Error
Trapezoid Rule	$1.11 \times 10^{-3}$
Simpson's 3/8 Rule	$4.41 \times 10^{-5}$
Gauss-Legendre	$1.452 \times 10^{-9}$
Gauss-Hermite	$1.619 \times 10^{-1}$
Clenshaw-Curtis	$2.557 \times 10^{-7}$

**Table 5: Comparison of Relative Errors for 7-point Quadrature Rules**

This is a summary of the relative errors for each integration technique applied to (4.5) with 7 evaluation points.

that this gave us

$$\int_1^4 \sqrt{x} \, dx \approx \frac{3}{2} \sum_{i=1}^n w_i \sqrt{1.5x_i + 2.5}. \quad (4.21)$$

Applying the Clenshaw-Curtis nodes and weights given in Figure 8, we obtain Table 4.

Adding up the last column of 4 and multiplying by  $\frac{dh}{dx} = \frac{3}{2}$ , we have

$$\frac{dh}{dx} \sum_{i=1}^n w_i \sqrt{h(x_i)} = \frac{3}{2} (3.11111190652774) = 4.66666785979161. \quad (4.22)$$

This result has an error of  $1.193 \times 10^{-6}$ , and a relative error of  $2.557 \times 10^{-7}$ .

Table 5 summarizes the relative errors we observed for each integration method.

## CHAPTER 5

### OVERVIEW OF INFERENCE-DRIVEN BRANCH AND BOUND

Suppose we have a global optimization problem in  $n$  variables,

$$\underset{x \in \Omega}{\text{minimize}} f(x) \text{ where } x = (x_1, x_2, \dots, x_n),$$

which satisfies the following two conditions:

- The solution space  $\Omega$  is **branchable**. That is, there is an operation  $\text{Branch}(S)$  by which some sets  $S \subseteq \Omega$  can be split into subsets called branches, as in Section 2.2, where elements of a branch share one or more properties that may affect solution fitness. Furthermore, each branch can again be divided into even smaller branches, within which elements share one or more additional properties. Ideally, this branching process can be recursed until some favorable subset of the branches contains few enough solutions that they can be evaluated directly.<sup>1</sup>
- The optimization problem is **structured**. By this, we mean that there is some correlation between the branch membership of optimal elements of  $\Omega$  and a parameter  $\gamma$  of the branch chosen so that it has an estimator  $\hat{\gamma}$  whose sampling distribution can be estimated from a relatively small sample of a branch. We will call  $\gamma$  the **guiding parameter**, and  $\hat{\gamma}$  the **guiding statistic**, because we will use them to decide where to look for optimal solutions.<sup>2</sup> Whether or not a problem is structured depends jointly on the solution space, the objective function  $f$ , and the choice of branching operation. We will adopt the convention that smaller values of  $\gamma$  are associated with higher likelihood of optimality; that is, the correlation between branch membership of optimal elements and  $\gamma$  is negative.<sup>3</sup>

---

<sup>1</sup>If this does not happen, random sampling of these branches may still tend to produce a more optimal result than a comparably sized random sample from the whole solution space. Likewise, local optimizations within these branches may also be fruitful.

<sup>2</sup>This requirement is to avoid “needle in a haystack” scenarios, where there is no correlation between the fitness of the optimal solution and that of other solutions – regardless of any similarities in their properties or composition.

<sup>3</sup>In a similar manner to the convention for objective functions, this choice of “smaller is better” comes with no loss of generality.

## 5.1 The Bounding Criterion

As we desire to follow a branch and bound pattern, and the branch operation is provided by our assumptions, the rest of this chapter is focused on devising a criterion for bounding. Our strategy for this is to use the guiding statistic  $\gamma$  to determine which branches to keep, and which others to discard, by following a simple rule: if  $A$  and  $B$  are branches and  $\gamma_A < \gamma_B$ , then  $B$  can be discarded. It is productive to think of  $\gamma_A$  as a proxy for the minimum of  $f(A)$ . As membership of optimal elements in a branch  $B$  is inversely correlated with  $\gamma_B$ , so that a smaller value of  $\gamma_B$  may indicate a larger concentration of optimal elements in  $B$ , then if this indication is not drowned out by noise from other factors that may affect the value of  $\gamma_B$ , we can find optimal elements efficiently by exploring only the branches with the smallest  $\gamma$  values so far observed.

## 5.2 Guiding Parameters and Statistics

Using this bounding criterion, if we could make decisions by letting  $\gamma_B$  be the minimal value of the objective function over all of the elements of a branch  $B$ , then eventual convergence to a set of branches containing the complete set of global optima would be guaranteed. After all, any branches containing a global optimum would have the lowest observable value of  $\gamma$  in any comparison, so every other branch could be discarded.<sup>4</sup>

Choosing to use the minimal objective function value over the branch as a guiding parameter requires evaluating the objective function for every element of the branch, as no sample smaller than the entirety of the branch can guarantee that we observe any information about the minimum. This ideal scenario demands that we perform an exhaustive search of the solution space. That is why we require that the error distribution of the estimating guiding statistic for a branch  $B$  must be able to be estimated from a random sample drawn from  $B$ . Intuitively, we want  $\gamma$  to be sensitive to a branch's minimum, and relatively insensitive to the less optimal elements of a branch. We want the parameter to behave like the branch minimum when used in our bounding criterion, while not requiring the whole branch to be evaluated to estimate its value.

The following are four examples of a reasonable choice for the guiding statistic. For each of these examples, let  $B \subseteq \Omega$  be any branch and let  $S$  be a sample consisting of  $n$  elements taken

---

<sup>4</sup>This is precisely the “ideal” bounding behavior we examined in Chapter 2.

from  $B$  uniformly at random with replacement.

### 5.2.1 Arithmetic Mean

The arithmetic mean works well when there are not relatively large objective function values present in each branch. The following logic is a direct application of the classic sampling distribution of the mean, the Central Limit Theorem, which was described briefly in Chapter 2. The guiding parameter is defined as the population mean,

$$\gamma = \mu_B = \frac{1}{|B|} \sum_{x \in B} x. \quad (5.1)$$

An unbiased estimate of  $\gamma$  can be computed from  $S$  by the formula for the sample mean,

$$\hat{\gamma} = \mu_S = \frac{1}{n} \sum_{x \in S} x. \quad (5.2)$$

For sufficiently large  $n$ , the distribution of the signed error of estimation  $\hat{\gamma} - \gamma$  is given by

$$\hat{\gamma} - \gamma = \mu_S - \mu_B \sim \mathcal{N}(0, \sigma_B^2/n). \quad (5.3)$$

Solving for the unknown parameter  $\gamma$ , this gives us a distribution for where  $\gamma$  is likely to be located based on an observation of  $\hat{\gamma}$ , by the relation

$$\gamma \sim \hat{\gamma} + \mathcal{N}(0, \sigma_B^2/n) = \mathcal{N}(\hat{\gamma}, \sigma_B^2/n). \quad (5.4)$$

It should be noted that how large the sample size  $n$  must be before it is “sufficiently large” is problem-specific. One may choose  $n$  to be appropriately large for a given problem by trial and error, or more robustly by increasing  $n$  for a given branch until its bootstrapped error distribution is passably normal under a distribution comparison such as the Lilliefors test [14].

The conventional heuristic is that  $n > 30$  is usually sufficient<sup>5</sup>.

---

<sup>5</sup>Some textbook authors prefer larger numbers than 30 for this, and the number tends to be somewhere between 30 and 80. Our sample sizes in practice are generally well in excess of these minimal sufficient sizes.

### 5.2.2 Small Quantiles

Elements of  $B$  that are relatively small under evaluation by  $f$  may be sufficiently close to the minimum of  $f(B)$  that their values can serve as a guiding statistic for some problems. For example, if the fitness of solutions in  $B$  is continuous with respect to some metric  $d$  defined on  $B$ , then it is intuitive that the solutions that have  $f$  scores close to the 1st percentile of  $f(B)$  will tend to be relatively close under  $d$  to solutions with the minimal  $f$  score. In this approach, we choose a number  $p \in (0, 0.5]$  and obtain an estimate of the  $p$ th quantile of  $f(B)$  from a sample  $S_0$  by computing the  $p$ th quantile of  $S_0$ . If  $X$  is the random variable such that  $f(B) \sim X$  under independent sampling from  $B$  with replacement, and  $F_X$  is the CDF of  $X$ , a convenient definition for the  $p$ th quantile of  $f(B)$  is

$$\Omega_p = F_X^{\leftarrow}(p). \quad (5.5)$$

A corresponding estimate of the  $p$ th quantile computed from  $S_0$  can be defined as

$$\hat{\Omega}_p = \hat{F}_S^{\leftarrow}(p), \quad (5.6)$$

where  $\hat{F}_S$  is the ECDF of the sample  $S$ . Note that the proportion  $p$  is effectively rounded down to the next  $k/n$  by  $\hat{F}_S$ , where  $k = 1, 2, \dots, n$ , and interpolation may be used to estimate quantiles between these values. Recall (3.7), which said that the error distribution of this estimate is asymptotically normal with mean 0. The variance, though given in terms of an unknown density  $f(\Omega_p)^2$ , can be estimated easily via bootstrap, as discussed in Section 3.6.

### 5.2.3 Sub-Quantile Means

This approach is a combination of the previous two. When using the arithmetic mean, large deviations affect  $\gamma$  more strongly than small ones, which makes it more sensitive to the minimum than to other negative deviations. However, this is equally true for both positive and negative deviations, so the effects of large positive deviations may dominate the value of  $\gamma$ . To avoid this, we can choose a small quantile  $p \in (0, 0.5]$ , and let  $\gamma_B$  be the arithmetic mean of the elements  $x \in B$  for which  $f(x)$  is less than the  $p$ th quantile of  $f(B)$ . This allows  $\gamma$  to ignore some of the more positive values in  $f(B)$  while remaining sensitive to extreme lower values - all at the

cost of requiring more points to achieve a sufficient sample size. Intuitively, the estimator  $\hat{\gamma}$  is given by the arithmetic mean of the sample elements  $s \in S$  such that  $f(s)$  is less than the  $p$ th quantile of  $f(S)$ . As in the previous method, the error of estimation is asymptotically normal, and the variance of the error distribution can be estimated by bootstrap. The asymptotic normality of the sampling distribution in this method is a direct consequence of the Central Limit Theorem, as this guiding statistic is the mean of a large sample from a subset of the population.

A straightforward procedure for sampling from this lower subset of the population is to choose a target sample size  $n$  and sample from the population until there are at least  $n$  elements below the  $p^{th}$  quantile of the sample. For large  $n$  and moderate  $p$ , the  $p^{th}$  quantile of the sample will converge to the  $p^{th}$  quantile of the population, which makes it an effective threshold for this purpose. The mean of the elements below the  $p^{th}$  sample quantile can then be used as the guiding statistic. Naturally, the number of sample points below the  $p^{th}$  population quantile in a sample of size  $n$  is well-modeled by a binomial CDF with success probability  $p$  and number of observations  $n$ .

#### 5.2.4 $1/n^{th}$ Quantile and the Sample Minimum

The  $1/n^{th}$  quantile of the population is a particularly strong choice of guiding parameter, as it converges to the population minimum as  $n \rightarrow \infty$ . The sample minimum is a biased estimator of the  $1/n^{th}$  quantile. Unlike the previous three guiding parameters we discussed, estimating the distribution of the error of estimation cannot be achieved easily by bootstrap without knowledge of the distribution of the population. We discuss why this issue arises in Section 7.1, and develop an effective method of estimating the error distribution in the rest of Chapter 7. The eventual result is a GEV model that approximates the error distribution  $\hat{\mathcal{Q}}_0 - \mathcal{Q}_{1/n}$ . As the guiding statistic  $\mathcal{Q}_0$ , the sample minimum, is a single point, it is very sensitive to errors in the computation or measurement of the objective function  $f$ . Future research on the robustness of methods using this guiding statistic may shed light on this issue.

### 5.3 Inputs and Outputs

The inference-driven branch and bound algorithm described in this thesis can be used as a black-box solver for a variety of optimization problems. An implementation is available on

GitHub [16]. For fast, high quality pseudo-random numbers, we utilize Blackman and Vigna’s wonderful Xoshiro256\*\* algorithm [3].

As input, the user must supply the following:

- Define a class inheriting from **Branch** that represents the solution space with a **GetBranches()** method and a **Sample()** method.
- Provide a fitness function that takes an element from a branch as an input, and returns a **double**.
- Select a guiding parameter, a sample size, and a confidence level for use in discarding branches.

Example **Branch** classes for intervals of the real line and rectangular regions of  $\mathbb{R}^2$  are provided for use with the test functions in Chapter 8, as well as a **PartialEMSPanBranch** class that allows our optimization algorithm to interface with the ambulance simulation described in Chapter 9.

The output is the set of branches  $\mathcal{S}$  after the last bounding step. These branches constitute a reduction of the search region, on which other optimization routines can then be run. A likely use case may be to begin with a problem that has  $10^{18}$  solutions, use branch and bound to reduce the search space to around  $10^3$  solutions, and then perform an exhaustive search on that reduced search space. Our implementation additionally keeps track of the best solution observed throughout the optimization process, and this can be used instead of another optimization routine if desired.



## CHAPTER 6

### COMPARING BRANCHES

#### 6.1 Overview

Let  $B_1, B_2, \dots, B_n$  be branches, and denote the guiding parameter of  $B_i$  by  $\gamma_i$ . Similarly, let  $\hat{\gamma}_i$  be a guiding statistic computed from a large sample from  $B_i$ , and let  $Y_i$  be the error distribution  $Y_i = \hat{\gamma}_i - \gamma_i$ . The previous section described how to compute  $\hat{\gamma}_i$  and estimate  $Y_i$  for each branch. This gives us a method of estimating the likelihood of each potential value of the guiding parameter  $\gamma_i$  for each branch by the relation

$$\gamma_i \sim \hat{\gamma}_i - Y_i \tag{6.1}$$

We want to use this information to decide which branches to discard, and which to branch further. Recall that our bounding criterion tells us to discard a branch  $B_j$  whenever there is another branch  $B_k$  with a preferable guiding parameter; that is, whenever  $\gamma_k < \gamma_j$ . Since  $Y_j$  and  $Y_k$  are random variables, we generally cannot be absolutely certain that  $\gamma_k < \gamma_j$ ; instead, we must estimate the probability with which that inequality holds. Let  $D_i$  be the event in which we discard the branch  $B_i$ . Then

$$P(D_i) = P(\gamma_1 < \gamma_i \cup \gamma_2 < \gamma_i \cup \dots \cup \gamma_{i-1} < \gamma_i \cup \gamma_{i+1} < \gamma_i \cup \dots \cup \gamma_n < \gamma_i). \tag{6.2}$$

This can be expressed more concisely in complement form, where the probability of keeping the branch  $B_i$ , given by  $1 - P(D_i)$ , is simply the probability that  $\gamma_i$  is the smallest value of  $\gamma$  among all our branches. Hence

$$1 - P(D_i) = P\left(\gamma_i \leq \min_{j \neq i} \gamma_j\right). \tag{6.3}$$

If the error distributions  $Y_1, Y_2, \dots, Y_n$  are assumed to be continuous, then  $P(\gamma_j = x) = 0$  for any real value  $x$ .<sup>1</sup> Since the samples were taken independently, it follows that the error distributions

---

<sup>1</sup> As the solution space is assumed to be very large, a continuous approximation of the error distributions will be reasonable unless the objective function is insensitive to small changes in the input. This is one of several reasons not to use a boolean indicator as an objective function.

$Y_1, Y_2, \dots, Y_n$  are mutually independent. Then we have

$$P(\gamma_j = \gamma_k) = P(\gamma_j = x \cap \gamma_k = x) = 0, \quad (6.4)$$

so the probability that two branches share the lowest value of  $\gamma$  is negligible. That means exactly one branch must have the lowest  $\gamma$ , so the set of events  $\{D_1^c, D_2^c, \dots, D_n^c\}$  forms a partition of the possible outcomes. For this reason, the complement discard probabilities have the convenient property of summing to unity. That is,

$$\sum_{i=1}^n (1 - P(D_i)) = 1. \quad (6.5)$$

Three of the four guiding statistics we have described have error distributions that are asymptotically normal. In addition to two more general methods, we give special consideration to the case where the sampling distributions of the guiding statistics are all normal.

## 6.2 Comparison by Monte Carlo Estimation of Proportion

Perhaps the simplest and most general method for computing the probability that each branch has the smallest guiding parameter is a direct, frequentist approach. The probability with which a branch  $B_i$  contains the lowest value of  $\gamma$  can be approximated by the long-run proportion of observations in which  $\gamma_i$  is the smallest element of a set of point samples taken from the estimated distributions of  $\gamma$  for each branch. Specifically, for observations  $j = 1, 2, \dots, m$ , we draw a point sample from each distribution (6.1) describing the position of  $\gamma_i$  for every branch index  $i = 1, 2, \dots, n$ . This gives us a list of simulated parameter values  $[\gamma_1^j, \gamma_2^j, \dots, \gamma_n^j]$  for the  $j^{th}$  observation ordered by their branch indices. The number of times each branch is associated with the smallest simulated value of  $\gamma$  in an observation  $j$  is counted over all  $m$  observations, and the resulting counts are divided by  $m$  to produce estimates of the proportion of time that each branch will produce the lowest value of  $\gamma$ . As  $m \rightarrow \infty$ , this proportion converges to the complement discard probability  $1 - P(D_i)$  for each branch  $B_i$ . While slow, this method is easy to implement and useful for sanity-checking other computations.

### 6.3 Pairwise Comparisons

In many cases, it may be expedient to consider pairwise comparisons between two branches - computing the probability of discarding one branch in light of exactly one other branch, rather than evaluating it with respect to the entire set of branches  $\mathcal{S}$ . Intuitively, since  $1 - P(D_i)$  is the probability that a branch has the lowest guiding parameter, and each additional branch considered is another competitor for this rank, pairwise comparisons necessarily give an overestimate of  $1 - P(D_i)$  and a corresponding underestimate of  $P(D_i)$ . Bounding performed based on such an estimate will therefore be more conservative - but not less reliable - than when using the entire set of branches. Symbolically, by applying the inequality  $P(A) \leq P(A \cup B)$  for any events  $A$  and  $B$  to (6.2), we can express this as

$$P(D_i) \geq P(\gamma_j < \gamma_i) \text{ for all } j \neq i. \quad (6.6)$$

As this decreases our certainty in discarding  $B_i$ , we seek to minimize this loss by choosing to keep the element of the union that gives us the greatest lower bound on  $P(D_i)$ , as

$$P(D_i) \geq \max_{j \neq i} P(\gamma_j < \gamma_i). \quad (6.7)$$

Let the event that  $\gamma_j < \gamma_i$  for a branch  $B_j$  be denoted by  $D_{i,j}$ . Then the previous inequality may be written as

$$P(D_i) \geq \max_{j \neq i} P(D_{i,j}). \quad (6.8)$$

#### 6.3.1 Guiding Statistics with Normally Distributed Errors

If the error distributions of the parameter estimates  $Y_i$  are normally distributed, we have  $\gamma_i \sim \mathcal{N}(\hat{\gamma}_i, \sigma_i^2)$  for each branch  $B_i$ . Observe that

$$P(D_{i,j}) = P(\gamma_j < \gamma_i) = P(\gamma_j - \gamma_i < 0). \quad (6.9)$$

By the result in Section 3.3, since a linear combination of independent normally distributed random variables is normally distributed, we have

$$Z := \gamma_j - \gamma_i \sim \mathcal{N}(\hat{\gamma}_j - \hat{\gamma}_i, \sigma_i^2 + \sigma_j^2). \quad (6.10)$$

Then we can rewrite (6.9) as

$$P(D_{i,j}) = P(Z < 0) = F_Z(0). \quad (6.11)$$

There are  $\binom{n}{2}$  possible comparisons among the  $n$  branches, and computing one normal CDF for each comparison to find  $\max_{i \neq j} P(D_{i,j})$  is an  $O(n^2)$  process in terms of the number of special function evaluations required. The number of necessary comparisons can be reduced by first finding two sets of non-dominated branches:

- The set  $L \subseteq \mathcal{S}$  of branches which are Pareto optimal under the minimization of both their means and variances.
- The set  $U \subseteq \mathcal{S}$  of branches which are Pareto optimal under maximization of their means and minimization of their variances.

We then have only to find the largest  $P(D_{i,j})$  where  $i$  comes from  $U$  and  $j$  comes from  $L$ . To see why this is true, we will need Theorem 6.3, which provides us with a useful result for comparing normal distributions.

**Lemma 6.1.** *If  $A, B$ , and  $C$  are independent normally distributed random variables with  $\mu_B < \mu_C$  and  $\sigma_B^2 = \sigma_C^2$ , then  $P(A < B) < P(A < C)$ .*

*Proof.* Suppose  $\mu_B < \mu_C$  and  $\sigma_B^2 = \sigma_C^2$ . Since  $A, B$ , and  $C$  are independent and normal, by the result in Section 3.3, any linear combination of them will be normally distributed. Let  $D = A - B$  so that  $D \sim \mathcal{N}(\mu_A - \mu_B, \sigma_A^2 + \sigma_B^2)$  and let  $E = A - C$  so that  $E \sim \mathcal{N}(\mu_A - \mu_C, \sigma_A^2 + \sigma_C^2)$ . Observe that since  $\mu_D - \mu_E = \mu_A - \mu_B - (\mu_A - \mu_C) = \mu_C - \mu_B$ , we have  $f_E(x) = f_D(x + (\mu_C - \mu_B))$ ,

because

$$\begin{aligned}
f_E(x) &= \frac{1}{\sqrt{2\pi\sigma_E^2}} e^{-\frac{(x-\mu_E)^2}{2\sigma_E^2}} \\
&= \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\frac{(x+\mu_D-\mu_E-\mu_D)^2}{2\sigma_D^2}} \\
&= \frac{1}{\sqrt{2\pi\sigma_D^2}} e^{-\frac{(x+\mu_C-\mu_B-\mu_D)^2}{2\sigma_D^2}} \\
&= f_D(x + (\mu_C - \mu_B)).
\end{aligned}$$

Then since  $\mu_B < \mu_C$ , we have  $\mu_C - \mu_B > 0$ , so

$$\begin{aligned}
F_E(0) &= \int_{-\infty}^0 f_E(x) dx \\
&= \int_{-\infty}^0 f_D(x + (\mu_C - \mu_B)) dx \\
&= \int_{-\infty}^{\mu_C - \mu_B} f_D(x) dx \\
&= \int_{-\infty}^0 f_D(x) dx + \int_0^{\mu_C - \mu_B} f_D(x) dx,
\end{aligned}$$

and as  $f_D(x) > 0$  for all  $x \in \mathbb{R}$ , we have

$$F_E(0) = \int_{-\infty}^0 f_D(x) dx + \int_0^{\mu_C - \mu_B} f_D(x) dx > \int_{-\infty}^0 f_D(x) dx = F_D(0).$$

Then since  $F_E(0) > F_D(0)$ , we have  $P(A - C < 0) > P(A - B < 0)$ , therefore

$P(A < C) > P(A < B)$  as desired. □

**Lemma 6.2.** *If  $A, B$ , and  $C$  are independent normally distributed random variables with  $\mu_B = \mu_C$  and  $\sigma_B^2 > \sigma_C^2$ , then  $P(A < B) < P(A < C)$  if and only if  $\mu_A < \mu_B$ .*

*Proof.* Suppose that  $\mu_B = \mu_C$  and  $\sigma_B^2 > \sigma_C^2$ . As in the previous lemma, since  $A, B$ , and  $C$  are independent and normal, any linear combination of them will be normally distributed. Let  $D = A - B$  so that  $D \sim \mathcal{N}(\mu_A - \mu_B, \sigma_A^2 + \sigma_B^2)$  and let  $E = A - C$  so that

$E \sim \mathcal{N}(\mu_A - \mu_C, \sigma_A^2 + \sigma_C^2)$ . Since  $\mu_B = \mu_C$ , we know  $\mu_D = \mu_A - \mu_B = \mu_A - \mu_C = \mu_E$ . Also, as  $\sigma_B^2 > \sigma_C^2$ , we have  $\sigma_D^2 = \sigma_A^2 + \sigma_B^2 > \sigma_A^2 + \sigma_C^2 = \sigma_E^2$ . Note that since  $\sigma_E^2$  and  $\sigma_D^2$  are also both non-negative, we can say  $\sigma_E < \sigma_D$ . Let  $Z_D = \frac{D - \mu_D}{\sigma_D}$  and  $Z_E = \frac{E - \mu_E}{\sigma_E}$  so that  $Z_D$  and  $Z_E$  both follow a standard normal distribution. Then

$$F_D(0) = \int_{-\infty}^0 f_D(x)dx = \int_{-\infty}^{\frac{-\mu_D}{\sigma_D}} f_{Z_D}(z)dz = \Phi\left(\frac{-\mu_D}{\sigma_D}\right),$$

and by the same logic,  $F_E(0) = \Phi\left(\frac{-\mu_E}{\sigma_E}\right) = \Phi\left(\frac{-\mu_D}{\sigma_E}\right)$ . Since the standard normal CDF  $\Phi(x)$  is a strictly increasing function, we know that  $a < b$  if and only if  $\Phi(a) < \Phi(b)$  for any real numbers  $a$  and  $b$ . Suppose  $\mu_A < \mu_B$ , so that  $\mu_E = \mu_D < 0$ . Then the quantities  $\frac{-\mu_D}{\sigma_D}$  and  $\frac{-\mu_D}{\sigma_E}$  must be positive. Then since  $\sigma_E < \sigma_D$ , we have  $\frac{1}{\sigma_E} > \frac{1}{\sigma_D}$  so  $\frac{\mu_D}{\sigma_E} < \frac{\mu_D}{\sigma_D}$  and  $\frac{-\mu_D}{\sigma_E} > \frac{-\mu_D}{\sigma_D}$ . Then  $\Phi\left(\frac{-\mu_D}{\sigma_E}\right) > \Phi\left(\frac{-\mu_D}{\sigma_D}\right)$  therefore  $F_E(0) > F_D(0)$ . Then  $P(A - C < 0) > P(A - B < 0)$ , so  $P(A < C) > P(A < B)$  as desired. Suppose instead that  $P(A < C) > P(A < B)$ . Then  $P(A - C < 0) > P(A - B < 0)$ , so  $F_E(0) > F_D(0)$  and  $\Phi\left(\frac{-\mu_D}{\sigma_E}\right) > \Phi\left(\frac{-\mu_D}{\sigma_D}\right)$ . As  $\Phi$  is strictly increasing, we have  $\frac{-\mu_D}{\sigma_E} > \frac{-\mu_D}{\sigma_D}$  and since  $\sigma_E < \sigma_D$  this is only possible if  $\mu_D < 0$ . Thus  $\mu_D = \mu_A - \mu_B$ , so  $\mu_A < \mu_B$  as desired.  $\square$

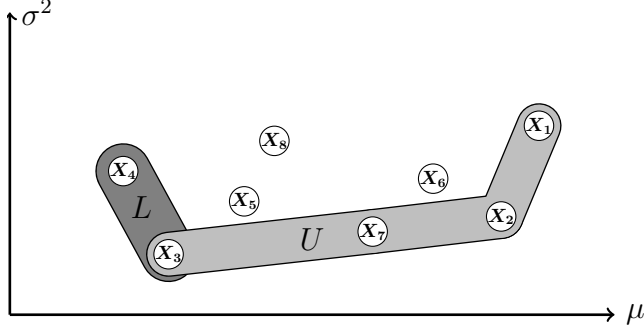
**Theorem 6.3.** *If  $A, B$ , and  $C$  are independent normally distributed random variables with  $\mu_A < \mu_B$  and  $\mu_A < \mu_C$ , then if any of the following conditions hold,*

(a)  $\mu_B < \mu_C$  and  $\sigma_B^2 \geq \sigma_C^2$  or

(b)  $\mu_B \leq \mu_C$  and  $\sigma_B^2 > \sigma_C^2$ ,

then  $P(A < B) < P(A < C)$ .

*Proof.* Suppose (a) holds, so  $\mu_B < \mu_C$  and  $\sigma_B^2 \geq \sigma_C^2$ . Let  $D$  be a normally distributed random variable independent from  $A$  with mean  $\mu_B$  and variance  $\sigma_C^2$ . Then by Lemma 6.2, we have  $P(A < D) > P(A < B)$ , and by Lemma 6.1 we have  $P(A < C) > P(A < D)$ . Therefore  $P(A < C) > P(A < B)$ . Suppose instead that (b) holds, so  $\mu_B \leq \mu_C$  and  $\sigma_B^2 > \sigma_C^2$ . Let  $E$  be a normally distributed random variable independent from  $A$  with mean  $\mu_C$  and variance  $\sigma_B^2$ . Then



**Figure 9: Example of the Sets  $L$  and  $U$ .**

This is an example of  $L$  and  $U$  for normally distributed parameter distributions  $X_1, X_2, \dots, X_8$ . Their union forms a convex “basket” shape holding all of the other solutions.

by Lemma 6.1, we have  $P(A < E) > P(A < B)$ , and by Lemma 6.2,  $P(A < C) > P(A < E)$ .

Therefore  $P(A < C) > P(A < B)$ . In every case, the result holds.  $\square$

Let  $X_i = \mathcal{N}(\hat{\gamma}_i, \sigma_i^2)$  be the estimated distribution of the guiding parameter  $\gamma_i$  for a branch  $B_i$ . Applying Theorem 6.3 to the set  $\{X_1, X_2, \dots, X_n\}$ , we have that if for some integers  $j$  and  $k$ ,  $\gamma_j \leq \gamma_k$  and  $\sigma_j^2 \leq \sigma_k^2$ , then  $P(D_{i,j}) \geq P(D_{i,k})$  for any choice of  $i$ . This means that when searching for the largest pair-wise discard probability  $P(D_{i,j})$ , if a distribution has smaller mean or variance, it will be a stronger choice for the distribution  $X_j$  against which  $X_i$  will be compared. Likewise, if there are integers  $h$  and  $i$  for which  $\gamma_i \geq \gamma_h$  and  $\sigma_i^2 \leq \sigma_h^2$ , then  $P(D_{i,j}) \geq P(D_{h,j})$  for any choice of  $j$ . Distributions with larger means and smaller variances will therefore be stronger choices for the distribution  $X_i$  in our comparison. This allows us to efficiently find the strongest pairwise comparison among all possible pairs of distributions by comparing elements of  $U$  to elements of  $L$ :

$$\max_{i \neq j} P(D_{i,j}) = \max \{P(D_{i,j}) \mid i \in U, j \in L\}. \quad (6.12)$$

An example of this is given by Figure 9.

#### 6.4 Comparisons Involving More Than Two Branches

In this section, we develop and refine a general method for computing discard probabilities given parameter distributions for each of the branches.

### 6.4.1 Derivation of the Discard Probability Integral

The computation of  $P(D_i)$  for a branch  $B_i$  requires us to compare it against all of the other branches simultaneously, as any one of them might have a better guiding parameter. Let  $X_i := \hat{\gamma} - Y_i$  be the parameter distribution for the branch  $B_i$  estimated from the sample  $S_0^i$ , so that  $\gamma_i \sim X_i$  by (6.1). Taking the complement of (6.2) and applying De Morgan's law, we have

$$1 - P(D_i) = P(\gamma_1 \geq \gamma_i \cap \gamma_2 \geq \gamma_i \cap \cdots \cap \gamma_{i-1} \geq \gamma_i \cap \gamma_{i+1} \geq \gamma_i \cap \cdots \cap \gamma_n \geq \gamma_i). \quad (6.13)$$

Using our definition for  $X_i$ , we can rewrite this as

$$1 - P(D_i) = P(X_i < X_1 \cap X_i < X_2 \cap \cdots \cap X_i < X_{i-1} \cap X_i < X_{i+1} \cap \cdots \cap X_i < X_n), \quad (6.14)$$

where the cases in which  $X_i = X_j$  for  $j \neq i$  have been disregarded, as the probability of this is vanishingly small. This is in agreement with (6.3) and (6.4).

Since the parameter distributions  $X_i$  are independent<sup>2</sup>, the joint probability is simply the product of the individual probabilities,

$$1 - P(D_i) = \prod_{j \neq i} P(X_i < X_j). \quad (6.15)$$

Another way of looking at this product takes an approach similar to the usual derivation of extreme order statistics, which is given by Wackerly [24]. Let  $M_i = \min_{j \neq i} X_j$ , and denote the CDF and PDF of  $X_k$  by  $F_k$  and  $f_k$ , respectively, for  $k = 1, 2, \dots, n$ . Then

$$P(M_i > x) = P(X_1 > x \cap X_2 > x \cap \cdots \cap X_{i-1} > x \cap X_{i+1} > x \cap \cdots \cap X_N > x) \quad (6.16)$$

$$= P(X_1 > x)P(X_2 > x) \cdots P(X_{i-1} > x)P(X_{i+1} > x) \cdots P(X_N > x) \quad (6.17)$$

$$= \prod_{j \neq i} P(X_j > x) = \prod_{j \neq i} (1 - F_j(x)). \quad (6.18)$$

---

<sup>2</sup>The independence of these distributions comes from the pairwise independence of the error distributions  $Y_i$ ,  $i = 1, 2, \dots, n$ , which in turn comes from our sampling procedure for obtaining  $S_0^i$ .



By (6.3), we can say  $1 - P(D_i) = P(X_i < M_i)$ . Conditioning on  $X_i$ , we have

$$1 - P(D_i) = \int_{R_{X_i}} P(X_i = x)P(M_i > x \mid X_i = x) dx \quad (6.19)$$

where  $R_{X_i}$  is the support of  $X_i$ . Since  $X_i$  is pairwise independent with  $X_j$  when  $j \neq i$ , we know that  $X_i$  and  $M_i$  are independent. Then we have

$$\int_{R_{X_i}} P(X_i = x)P(M_i > x \mid X_i = x) dx = \int_{R_{X_i}} P(X_i = x)P(M_i > x) dx \quad (6.20)$$

$$= \int_{R_{X_i}} f_i(x) \prod_{j \neq i} (1 - F_j(x)) dx. \quad (6.21)$$

Thus

$$1 - P(D_i) = \int_{-\infty}^{\infty} f_i(x) \prod_{j \neq i} (1 - F_j(x)) dx. \quad (6.22)$$

This tells us exactly how to compute the probability of discarding  $B_i$ . We simply need to know the PDF  $f_i$  and the CDFs  $F_j$  for  $j \neq i$ , which were determined for each of our four examples of guiding parameters in Section 5.2.

#### 6.4.2 Alternative Forms for The Discard Probability Integral

For each of our four guiding statistics, the integrand (6.22) is continuous on  $\mathbb{R}$  and therefore integrable. However, evaluation of the integral may not be tractable analytically. Instead, we do that numerically using the quadrature techniques in Chapter 4. This exposes our calculations to some issues that are common in numerical methods, most notably the need to minimize the required computation time and the cumulative effects of representation error on the result.

The accumulation of representation error can be largely mitigated by eliminating all of the subtractions from the product. To achieve this, we define negated parameter distributions  $N_j = -X_j$  for each  $j = 1, 2, \dots, n$ . Then  $D_i$  occurs whenever the observation from  $N_i$  is less than

the observation from  $N_j$  for some  $j \neq i$ , which means

$$1 - P(D_i) = P(N_i > \max_{j \neq i} N_j). \quad (6.23)$$

Define  $M_i = \max_{j \neq i} N_j$ . Following the usual derivation of the distribution of the maximum, we have

$$P(M_i < x) = P(N_1 < x \cap N_2 < x \cap \cdots \cap N_{i-1} < x \cap N_{i+1} < x \cap \cdots \cap N_n < x) \quad (6.24)$$

$$= P(N_1 < x)P(N_2 < x) \cdots P(N_{i-1} < x)P(N_{i+1} < x) \cdots P(N_n < x) \quad (6.25)$$

$$= \prod_{j \neq i} P(N_j < x) \quad (6.26)$$

$$= \prod_{j \neq i} F_{N_j}(x). \quad (6.27)$$

Conditioning on the value of  $N_i$ , we have

$$P(M_i < N_i) = \int_{R_{N_i}} P(N_i = x)P(M_i < x \mid N_i = x) dx. \quad (6.28)$$

Since  $X_1, X_2, \dots, X_n$  are pairwise independent, it follows that  $N_1, N_2, \dots, N_n$  are also independent.

Then we have

$$P(M_i < N_i) = \int_{R_{N_i}} P(N_i = x)P(M_i < x) dx. \quad (6.29)$$

Combining this with (6.23) and (6.27) gives us

$$1 - P(D_i) = \int_{-\infty}^{\infty} f_{N_i}(x) \prod_{j \neq i} F_{N_j}(x) dx. \quad (6.30)$$

Quadrature computations using this integral are less prone to accumulating representation errors than those which use (6.22).

### 6.4.3 Fast Computation of the Discard Probability Integral

To compute the discard probability  $P(D_i)$  for each branch  $B_i$ , (6.22) and (6.30) will both require an evaluation of the PDF of  $N_i$  and the CDFs of the remaining  $n - 1$  distributions, for a

total of  $n$  special function calls per evaluation point. The task of computing all of the discard probabilities for the complete set of branches  $B_1, B_2, \dots, B_n$  using a quadrature rule with  $m$  evaluation points will therefore involve  $mn^2$  special function calls. Significant time savings may be obtained by altering the integrand to contain a product that is invariant over each of the discard probability computations, as in

$$1 - P(D_i) = \int_{-\infty}^{\infty} \frac{f_{N_i}(x)}{F_{N_i}(x)} \prod_{j=1}^n F_{N_j}(x) dx. \quad (6.31)$$

Applying the general quadrature approximation (4.4), with  $m$  evaluation points, we have

$$1 - P(D_i) \approx \sum_{k=1}^m w_k \frac{f_{N_i}(x_k)}{F_{N_i}(x_k)} \prod_{j=1}^n F_{N_j}(x_k). \quad (6.32)$$

On the first use of an evaluation point  $x_k$ , a total of  $n + 1$  special function calls are performed, and the value of the weighted product of CDFs  $w_k \prod_{j=1}^n F_{N_j}(x_k)$  is stored. Every subsequent use of that evaluation point will require only one or two special function calls, depending on whether or not the individual CDF values are stored as well. If a sufficiently broad interval of integration that covers all of the intervals  $R_{N_j}$ ,  $j = 1, 2, \dots, n$ , can be determined, then a subset of the same  $m$  evaluation points can be reused for all  $n$  discard probability computations by integrating over that same interval each time. Instead of  $mn^2$  special function evaluations, only  $mn$  are required.

#### 6.4.4 Determining the Interval of Integration

If pairwise discard probabilities can be computed efficiently for the guiding statistic used, we can pre-emptively discard any branch  $B_i$  for which  $1 - P(D_{i,j}) < \epsilon_M$  for some other branch  $B_j$ , where  $\epsilon_M$  is the unit roundoff error of the number system with which we compute the discard probabilities. The actual discard probability  $P(D_i)$  will be greater than  $P(D_{i,j})$ , as described in Section 6.3, so we can confidently discard these branches preemptively without it meaningfully affecting the total probability.<sup>3</sup> The parameter distributions for the subset of branches that were not preemptively discarded can then be used to construct a common interval

---

<sup>3</sup>The complement discard probabilities must sum to one by (6.5). In double precision arithmetic, with collections of  $10^k$  or fewer branches, there will be at least  $15 - k$  orders of magnitude between the scale of the most significant discard probabilities and the scale at which these preemptive discards affect the total probability.

of integration  $[a, b]$ . The lower bound of the interval is given by

$$a := \max_j \left\{ F_{N_j}^{\leftarrow}(\epsilon_M) \right\}, \quad j = 1, 2, \dots, k, \quad (6.33)$$

and the corresponding upper bound is

$$b := \max_j \left\{ F_{N_j}^{\leftarrow}(1 - \epsilon_M) \right\}, \quad j = 1, 2, \dots, k, \quad (6.34)$$

where the branches have been indexed so that  $B_1, B_2, \dots, B_k$  are the branches kept after preemptive discarding, and  $B_{k+1}, B_{k+2}, \dots, B_n$  are branches that were preemptively discarded during this pairwise step. For guiding statistics with normal sampling distributions given by  $N_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$ , the lower bound will be the largest of  $\mu_j - Z_{\epsilon_M} \sigma_j$  over all possible choices of  $j$ . Likewise, the upper bound will be the largest of  $\mu_j + Z_{\epsilon_M} \sigma_j$ . We believe stronger bounds can likely be obtained for many integrands (6.31), which would speed up the convergence of the quadrature computation.<sup>4</sup> This could merit further research, though our bounds (6.33) and (6.34) are sufficient for our purposes.

## 6.5 Comparison of Quadrature Methods for Computing Discard Probabilities

Each of the quadrature methods in Chapter 4 can be used to evaluate (6.30). In the case of the Newton-Cotes methods, the integral requires no alteration. For Gauss-Legendre and Clenshaw-Curtis, the integral must be transformed to be over the interval  $[-1, 1]$ , which is easily achieved by applying (4.3). In the case of Gauss-Hermite, if the guiding parameter estimates are normally distributed, it is a natural choice to use a change of variables  $z = \frac{x - \mu_i}{\sqrt{2}\sigma_i}$ , with

$dx = \sqrt{2}\sigma_i dz$  to get

$$f_i(x) = \frac{e^{-z^2}}{\sqrt{2\pi\sigma_i^2}}, \quad (6.35)$$

---

<sup>4</sup>Any evaluation points at which the integrand is less than  $\epsilon_M$  are wasted, as they will not contribute a meaningful amount of information to the discarding process, in which comparisons are made at a scale of around  $10^{-2}$ . These integrands are generally less than  $\epsilon_M$  over much of  $[a, b]$ , so computational savings could be obtained from tighter bounds.

which changes (6.30) into

$$1 - P(D_i) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-z^2} \prod_{j \neq i}^n F_j(\mu_i + \sqrt{2}\pi\sigma_i z) dz. \quad (6.36)$$

The Gauss-Hermite approximation of this integral with  $m$  evaluation points is given by

$$1 - P(D_i) \approx \frac{1}{\sqrt{\pi}} \sum_{k=1}^m w_k \prod_{j \neq i}^n F_j(\mu_i + \sqrt{2}\pi\sigma_i x_k). \quad (6.37)$$

This is sufficient for the computation of a single discard probability. However, since these  $x_k$  are  $z$ -values, and therefore depend on  $\mu_i$  and  $\sigma_i$  from the distribution  $N_i$ , this approach cannot be used to recycle evaluation points for efficient computation of discard probabilities for multiple branches. An alternative would be to define a Gaussian

$$g(x) = e^{-\left(\frac{x-c_1}{c_2}\right)^2}, \quad (6.38)$$

where  $c_1$  and  $c_2$  are chosen so that  $g(x) \geq \epsilon_M$  on  $[a, b]$ . This can be achieved by choosing  $c_1 = (a + b)/2$  and  $c_2 = (b - a)/(2Z_{\epsilon_M})$ . Multiplying the integrand in (6.31) by  $g(x)/g(x)$ , we have

$$1 - P(D_i) = \int_{-\infty}^{\infty} e^{-\left(\frac{x-c_1}{c_2}\right)^2} e^{\left(\frac{x-c_1}{c_2}\right)^2} \frac{f_{N_i}(x)}{F_{N_i}(x)} \prod_{j=1}^n F_{N_j}(x) dx. \quad (6.39)$$

Let  $z = \frac{x - c_1}{c_2}$ . Then we have

$$1 - P(D_i) = c_2 \int_{-\infty}^{\infty} e^{-z^2} e^{z^2} \frac{f_{N_i}(c_1 + c_2 z)}{F_{N_i}(c_1 + c_2 z)} \prod_{j=1}^n F_{N_j}(c_1 + c_2 z) dz. \quad (6.40)$$

The Gauss-Hermite approximation of this integral with  $m$  evaluation points is given by

$$1 - P(D_i) \approx c_2 \sum_{k=1}^m w_k e^{x_k^2} \frac{f_{N_i}(c_1 + c_2 x_k)}{F_{N_i}(c_1 + c_2 x_k)} \prod_{j=1}^n F_{N_j}(c_1 + c_2 x_k). \quad (6.41)$$

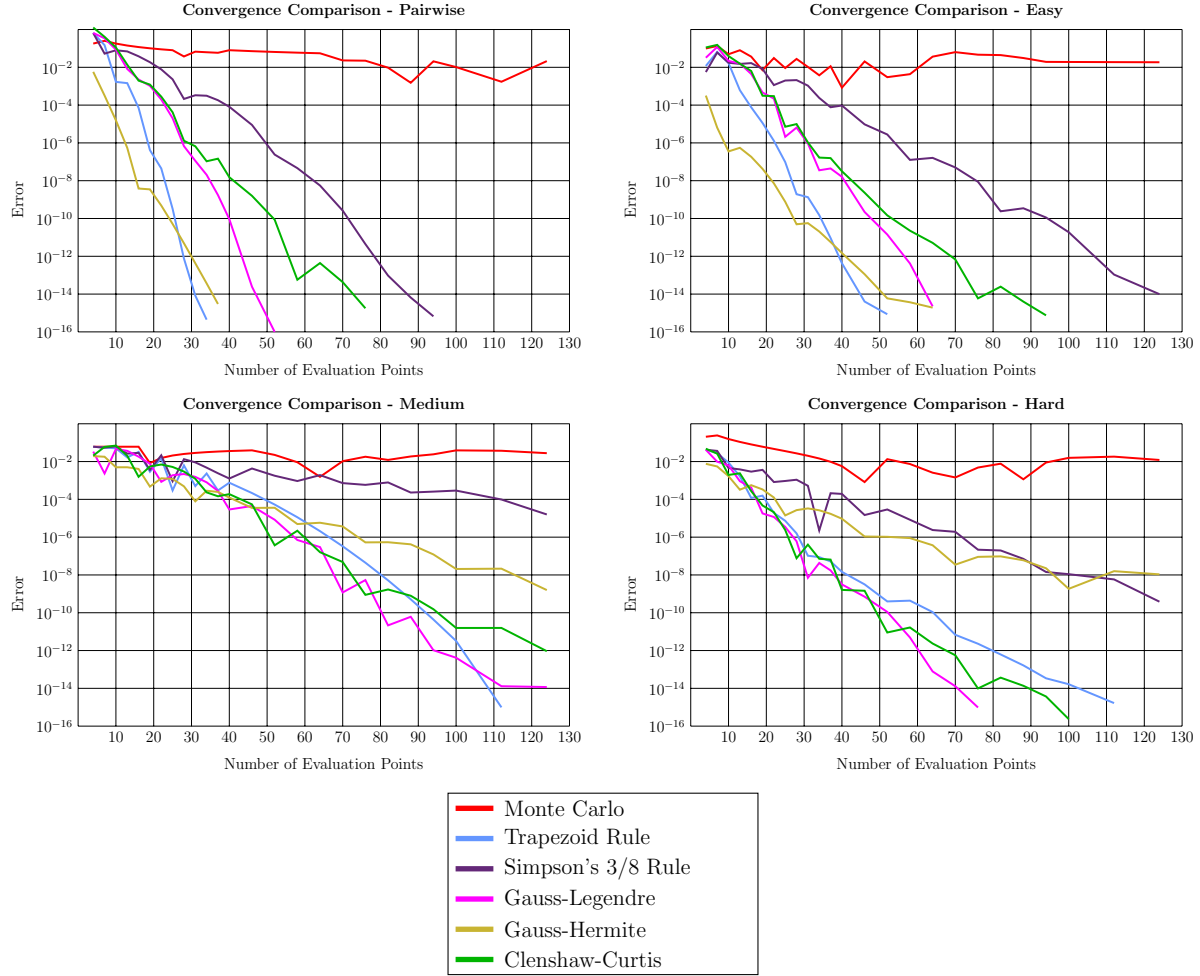
The values of  $c_2 w_k e^{x_k^2} \prod_{j=1}^n F_{N_j}(c_1 + c_2 x_k)$  can be computed once for each  $k = 1, 2, \dots, m$ , and then reused for every computation of  $P(D_i)$  for  $B_i \in \mathcal{S}$ . The CDF values  $F_{N_i}(c_1 + c_2 x_k)$  may also be recorded and reused at the cost of requiring significantly more storage space.

Figure 10 shows comparisons of per-evaluation performance for computing one discard probability for collections of branches with arrangements of parameter distributions that are progressively more difficult to integrate numerically. For the hardest and most realistic conditions, Gauss-Legendre and Clenshaw-Curtis are the most efficient, and the nesting property of Clenshaw-Curtis makes it an excellent choice for our implementation. The trapezoid rule is surprisingly efficient for this integral, and developing a well-optimized adaptive trapezoid rule implementation for this integral may be a direction for future research.

## 6.6 Discarding with a Confidence Level

We desire to discard as many branches as possible in each bounding step, while maintaining a high level of confidence that the set of branches we have not discarded contains the branch with the best value of the guiding parameter. Let  $\mathcal{S} = \{B_1, B_2, \dots, B_n\}$  be a set of branches indexed in decreasing order of  $P(D_i)$ , so that  $P(D_1) \geq P(D_2) \geq \dots \geq P(D_n)$ , and let  $\alpha \in (0, 1)$ . Recall from (6.3) that  $1 - P(D_i)$  can be interpreted as the probability that  $B_i$  has the best value of the guiding parameter  $\gamma$ . Furthermore, (6.5) states that these complement discard probabilities sum to unity. With these results in mind, when a branch  $B_j$  is discarded, it still has a probability of having had the best value of  $\gamma$  given by  $1 - P(D_j)$ , so our confidence that the collection of remaining branches  $\mathcal{S} \setminus \{B_k\}$  contains the branch with the best value of  $\gamma$  decreases by  $1 - P(D_j)$ . Let  $m$  be the largest number of branches that can be discarded while maintaining a confidence level  $1 - \alpha$  that the best branch is being kept. One way to obtain  $m$  is by first discarding the distribution that is least likely to have the best parameter value, which is  $B_1$ , then discarding the next least likely, which is  $B_2$ , and so on until the accumulated complement discard probability would exceed  $\alpha$  if any additional branches were discarded. In symbols, we define  $m$  to be

$$m = \max_k \left\{ \sum_{j=1}^k (1 - P(D_j)) \leq \alpha \right\}. \quad (6.42)$$



**Figure 10: Convergence Rate Comparison of Quadrature Methods**

These graphs give the logarithm of the error of each discard probability approximation by numbers of evaluation points used. All parameter distributions used were normal. “Pairwise” uses only two branches. “Easy” uses a set of 8 branches with distributions having  $\mu \in [1, 5]$  and  $\sigma \in [0.2, 1.2]$ . “Medium” uses a set of 20 branches with  $\mu \in [47, 90]$  and  $\sigma \in [0.3, 8]$ . “Hard” uses 120 branches with  $\mu \in [30, 120]$  and  $\sigma \sim |K| + 0.3$  where  $K \sim N(20, 15)$ .

The branches  $B_1, B_2, \dots, B_m$  are discarded, while  $B_{m+1}, B_{m+2}, \dots, B_n$  are retained for further examination.

In the situation that all the branches have very similar distributions of their guiding parameters, this definition of  $m$  may indicate that some branches should be discarded, even though they are all equally fit, if there are enough branches so that  $1/n < \alpha$ . However, if the parameter distributions cannot distinguish any branches as being preferable to others, we want to retain all of the branches with the hope that in later steps we can be more selective. This behavior can be achieved by choosing only to discard branches  $B_i$  such that  $P(D_i) \gg P(D_n)$ .

It should be noted that this computation applies to the set of branches under consideration only - not to those of any previous bounding step. Moreover, it makes comparisons using the probability with which each branch has the best guiding parameter value, and not necessarily a global minimum. As this is applied once for every branching step in the algorithm, our confidence may, in the worst case, decay exponentially with each iteration of branch and bound. In practice, however, keeping the best branch during each bounding step is not always an independent trial with probability of success  $(1 - \alpha)$ . When using the  $1/n^{th}$  quantile guiding parameter, for instance, if  $\mathcal{S}$  is converging toward better and better collections of branches, we can be more and more certain at later steps that our previous bounding decisions were good; that is, that the branches we previously discarded did not contain the best available value of the guiding parameter. This is because as  $\mathcal{S}$  improves in this way, the spread of the distribution of the guiding parameter for retained branches will decrease, offering a clearer picture of the fitness of the best solutions. Previous discards based on conservative estimates of the parameter distributions will therefore only become more certain in light of new data.

Better quantifying the confidence level over multiple steps of branch and bound is a topic for future research. For our present purpose, it is sufficient to set a confidence level and use it in each bounding step without consideration for how errors may compound over multiple steps.



## CHAPTER 7

### MODEL FITTING METHOD FOR EXTREME QUANTILE GUIDING PARAMETERS

#### 7.1 On Bootstrapping the Error Distribution of the Sample Minimum

Let  $X$  be a random variable, and suppose  $-X \in \mathcal{D}(G)$  for some GEV distribution  $G$  with parameters  $\mu$ ,  $\sigma$ , and  $\xi$ . Let  $S = \{X_1, X_2, \dots, X_n\}$  be a set of independent random variables with common distribution  $X$ . Then by the Fisher-Tippett-Gnedenko theorem,

$$M_n = \max_{i=1,2,\dots,n} (-X_i) \sim G \quad (7.1)$$

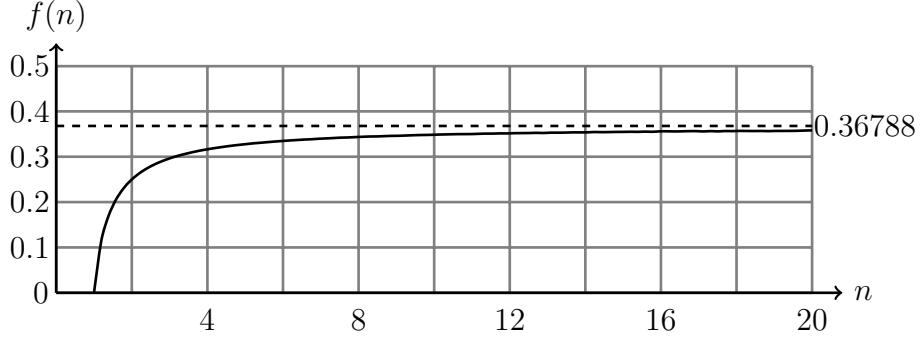
for sufficiently large  $n$ . Then the distribution of

$$m_n = \min_{i=1,2,\dots,n} (X_i) = -M_n \quad (7.2)$$

can be obtained from the CDF of  $G$ , denoted  $F_G(x)$ , by the relation

$$F_{m_n}(x) = 1 - F_{-m_n}(-x) = 1 - F_{M_n}(-x) = 1 - F_G(-x). \quad (7.3)$$

We can therefore model the distribution of the minimum  $m_n$  of a large sample taken from  $X$  by estimating  $G$ . Bootstrap estimation using  $S$  is a natural next step. However, if we observe the minima of bootstrap samples taken from  $S$ , then the distribution of these minima does not effectively model the distribution of sample minima taken from the population. Suppose we construct bootstrap samples  $S_b$  from  $S$  by drawing  $n$  observations from  $S$  uniformly at random with replacement. The probability of the minimum of  $S_b$  being the minimum of  $S$  is the probability that the minimum of  $S$  was observed during sampling, which can be modeled with a binomial distribution. We have  $n$  opportunities to observe the minimum of  $S$  during the construction of the bootstrap sample  $S_b$ , and in each opportunity we have probability  $1/n$  of selecting the minimum. As such, the probability of not selecting the minimum at all will be given by  $\left(\frac{n-1}{n}\right)^n$ , so the probability of observing the minimum of  $S$  during the construction of  $S_b$  is



**Figure 11: Graph of  $f(n) = \left(\frac{n-1}{n}\right)^n$  for  $n$  up to 20**  
The value of  $f$  rapidly converges to  $1/e \approx 37\%$  as  $n \rightarrow \infty$ .

exactly  $1 - \left(\frac{n-1}{n}\right)^n$ . This proportion converges to 63% of all bootstrap samples, regardless of the underlying distribution. Figure 11 demonstrates the rapid convergence of this proportion, even for very modest sample sizes.

A similar argument shows that of the remaining 37% of the samples, about 63% have the second smallest observation from the original sample as their minimum, and so on. For large  $n$ , this line of reasoning characterizes the bootstrapped distribution of the sample minimum in terms of the sample data without computing it. Furthermore, it tells us effectively nothing about the sampling distribution that we want to model.

The problem with this approach is that bootstrap cannot select values that are between adjacent sample elements or outside of the sample range. This restricts the variety of values that can be observed near the minimum, and imposes a binomial pattern for observing the sample minimum in each bootstrap sample. To get around this, we can replace the standard ECDF model of the population CDF that is used in bootstrap with a new model that provides a smoother approximation of the population's lower tail. Toward that end, we will apply the Pickands-Balkema-De Haan theorem to construct a model of the lower tail, and use method of moments to estimate the parameters of that model.

## 7.2 Estimating Upper Tail Moments

Let  $Y = -X$ , so that an estimate of the upper tail of  $Y$  is an estimate of the lower tail of  $X$  under a reflection. Let  $D = \{y_1, y_2, \dots, y_n\}$  be a sample drawn from  $Y$ , indexed so that  $y_i \leq y_{i+1}$  for  $i = 1, 2, \dots, n$ . Since  $Y \in \mathcal{D}(G)$ , by the Pickands-Balkema-De Haan theorem, there

exist  $u \in [y_0, y_n)$ ,  $a > 0$  and  $c \in \mathbb{R}$  such that for  $x > u$ ,

$$F_u(x) = \frac{F_Y(u+x) - F_Y(u)}{1 - F_Y(u)} \approx F_{GPD}(x; 0, a, c). \quad (7.4)$$

Define the “tail distribution”  $T$  by the CDF

$$F_T(x) = F_{GPD}(x; u, a, c) \quad (7.5)$$

so that  $T$  is a translated and rescaled form for our model of the upper tail of  $Y$ .

We can characterize  $u$  as the threshold beyond which the CDF of  $Y$  is well-modeled by a GPD.<sup>1</sup> Let the value of the model parameter  $u$  be given. Define  $Z = \{z_1, z_2, \dots, z_k\}$  to be the set of sample points  $y \in D$  such that  $y > u$ , indexed so that  $z_i \leq z_{i+1}$  for  $i = 1, 2, \dots, k$ . The ECDF of the points in the subset  $Z$  may be used as a model of the conditional excess distribution  $F_u(x)$ . Rather than using the standard ECDF, we will perform some modifications to improve the fit of the ECDF with a GPD model before using it to estimate the moments of the tail distribution with CDF  $F_T(x)$ .

The ECDF of  $Z$  specifies the locations at which the modeled CDF reaches 0 and 1 to be  $z_1$  and  $z_k$ , respectively. However, based on our choice of  $u$ , along with  $F_{GPD}$ , the model should reach 0 at  $u$  rather than  $z_1$ . Furthermore, depending on the shape parameter  $c$ , we may not want to specify that it reaches 1 at all.<sup>2</sup> Define a new CDF, denoted  $L(x)$ , by

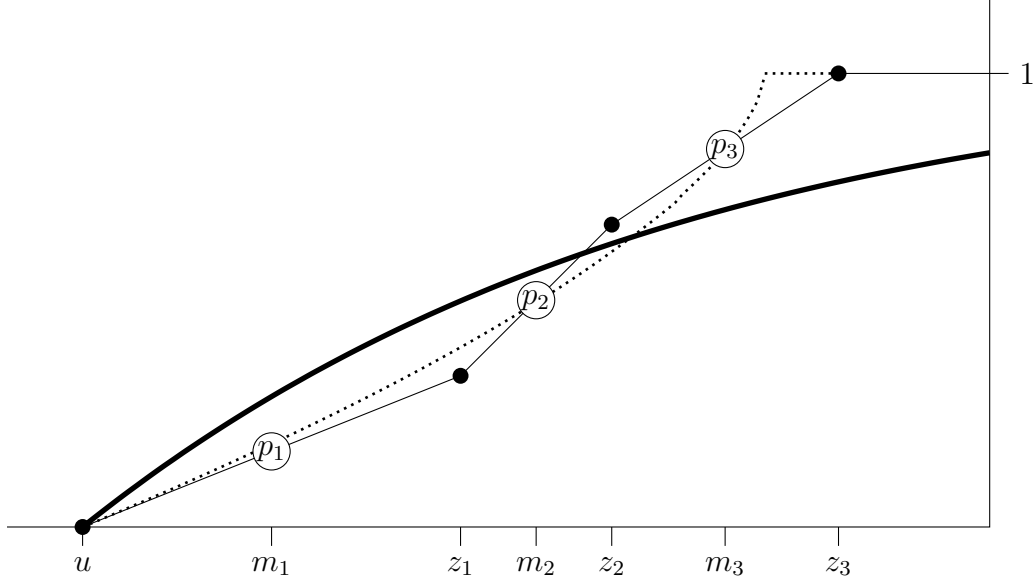
$$L(x) = \begin{cases} 0 & \text{if } x < u, \\ \frac{x-u}{k(z_1-u)} & \text{if } u \leq x < z_1, \\ \frac{i}{k} + \frac{x-z_i}{k(z_{i+1}-z_i)} & \text{if } z_i \leq x < z_{i+1} \text{ for } i \in \{1, 2, \dots, k-1\}, \text{ and} \\ 1 & \text{if } x \geq z_k. \end{cases} \quad (7.6)$$

This function  $L(x)$  linearly interpolates from the point  $(u, 0)$  to the top of the first step of the ECDF of  $Z$  at  $(z_1, 1/k)$ , then between the tops of the remaining steps over the interval  $[z_1, z_k]$ .

---

<sup>1</sup>Informally, we can think of  $u$  as being the location at which the tail of the distribution begins.

<sup>2</sup>If  $c \geq 0$ , the tail estimate is unbounded above, so specifying an  $x$  at which the CDF should be 1 is inappropriate.



**Figure 12: Graph of the Modified Excess ECDF for Fitting an Upper-Tail Estimate**  
This is a graph of the modified ECDF  $L(x)$  with 3 points  $z_1, z_2, z_3 > u$ . An MSE-fit GPD tail CDF is shown as a dotted line, and our conservative WMSE-fit tail CDF is given by the thick line.

An example of  $L(x)$  for a very small set  $Z$  is given in Figure 12. Note that our tail estimate is very conservative. Without the weighting, a purely MSE-fit tail may, for instance, assign zero probability density to  $z_k$ , which can be seen in Figure 12.

To estimate the first two moments of the tail CDF model, we will compute the exact first and second moments of a random variable  $\mathcal{L}$  with CDF given by  $L(x)$ . For the convenience of notation, let  $z_0 = u$ . The first moment of  $\mathcal{L}$  is given by

$$\begin{aligned}
E(\mathcal{L}) &= \int_{z_0}^{z_k} x \frac{dL}{dx} dx \\
&= \frac{1}{n} \sum_{i=1}^k \frac{1}{z_i - z_{i-1}} \int_{z_{i-1}}^{z_i} x dx \\
&= \frac{1}{n} \sum_{i=1}^k \frac{z_i^2 - z_{i-1}^2}{2(z_i - z_{i-1})} \\
&= \frac{1}{2n} \sum_{i=1}^k (z_i + z_{i-1}) \\
&= \frac{1}{n} \left( \frac{u + z_k}{2} + \sum_{i=1}^{k-1} z_i \right).
\end{aligned}$$

This is a weighted mean of the set  $Z \cup \{u\}$ , where the endpoints are given half of the usual unit weight. For the second moment, we have

$$\begin{aligned}
E(\mathcal{L}^2) &= \int_{z_0}^{z_k} x^2 \frac{dL}{dx} dx \\
&= \frac{1}{n} \sum_{i=1}^k \frac{1}{z_i - z_{i-1}} \int_{z_{i-1}}^{z_i} x^2 dx \\
&= \frac{1}{n} \sum_{i=1}^k \frac{z_i^3 - z_{i-1}^3}{3(z_i - z_{i-1})} \\
&= \frac{1}{3n} \sum_{i=1}^k (z_i^2 + z_i z_{i-1} + z_{i-1}^2) \\
&= \frac{1}{3n} \left( u^2 + z_k^2 + \sum_{i=1}^k z_i z_{i-1} + 2 \sum_{i=1}^{k-1} z_i^2 \right).
\end{aligned}$$

### 7.3 Estimating the Tail Distribution

Pickands provides an efficient method for estimating the model parameters  $u$ ,  $a$ , and  $c$  [10]. He uses a quantile-based argument to estimate  $a$  and  $c$  for a given value of  $u$ , and then optimizes over a large number of convenient choices of  $u$  - keeping the estimate that fits the ECDF of  $Z$  most closely under the  $L^\infty$  norm. The structure of the method we develop in this section is based on that approach.

In the previous section, given a value of  $u$ , we estimated the first two moments of the upper tail from the set of datapoints  $Z$ . To construct our GPD model of the upper tail of  $Y$  for a given  $u$ , we derive method of moments estimators for  $a$  and  $c$ , and plug in our values of  $E(\mathcal{L})$  and  $E(\mathcal{L}^2)$ .

The moment generating function of a GPD with location  $u$ , scale  $a$ , and shape  $c$  is given by

$$M(t) = \sum_{j=0}^{\infty} \left( \frac{(at)^j}{\prod_{k=0}^j (1 - kc)} \right), \quad kc < 1. \quad (7.7)$$

When we apply this model fitting to objective function values, since there are finitely many solutions in the search space, the tail of the distribution cannot be infinite. As finite tails correspond to negative values of the shape parameter  $c$ , we should expect  $c$  to be negative. Hence

$kc < 1$  is satisfied. Let  $P \sim GPD(u, a, c)$ . Using the convention that  $0^0 = 1$ , we obtain

$$E(P) = u + \frac{a}{1-c}, \quad (7.8)$$

$$E(P^2) = u^2 + 2u \frac{a}{1-c} + \frac{2a^2}{(1-c)(1-2c)}, \quad (7.9)$$

and

$$V(P) = \frac{a^2}{(1-c)^2(1-2c)}, \quad (7.10)$$

which agree with well-known results for the mean and variance of this distribution [12]. Setting  $E(P) = E(\mathcal{L})$  and  $E(P^2) = E(\mathcal{L}^2)$ , we obtain

$$\hat{c} = \min \left( \frac{1}{2} \left( 1 - \frac{(E(\mathcal{L}) - u)^2}{V(\mathcal{L})} \right), 0 \right) \quad (7.11)$$

and

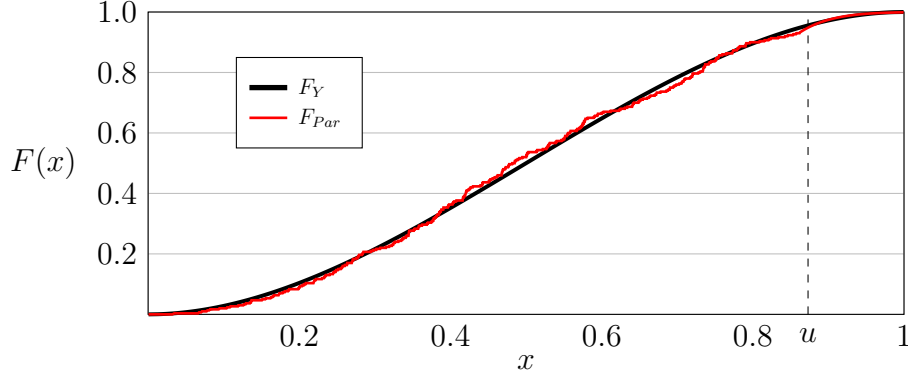
$$\hat{a} = (E(\mathcal{L}) - u)(1 - c) \quad (7.12)$$

where  $V(\mathcal{L}) = E(\mathcal{L}^2) - E(\mathcal{L})^2$ . As the population fitness is bounded below, we can assert that  $c < 0$ , which is why any positive estimate for  $\hat{c}$  is clamped to zero here.

Let  $\hat{F}_u$  be the CDF of a GPD with location  $u$ , scale  $\hat{a}$ , and shape  $\hat{c}$ . Let the midpoints of  $z_i$  be denoted  $m_i = (z_i + z_{i+1})/2$  for  $i = 1, 2, \dots, k-1$ . We construct points  $p_1, p_2, \dots, p_{k-1}$  given by  $p_i = (m_i, L(m_i))$ . For each of  $n/4$  choices of the threshold  $u$ , equally spaced over the interval  $[y_1, y_n]$ , we compute a figure of goodness of fit given by a weighted mean squared error evaluated against the points  $p_1, p_2, \dots, p_{k-1}$ ,

$$\text{WMSE}(u) = \frac{e^{5|c|}}{n} \sum_{i=1}^{k-1} \left( \hat{F}_u(m_i) - L(m_i) \right)^2. \quad (7.13)$$

The weight  $e^{5|c|}$  penalizes having a large magnitude in the shape parameter. This encourages conservative estimation of the tail thickness. The value of  $u$  that minimizes the WMSE becomes the estimate  $\hat{u}$ , coupled with its associated values of  $\hat{a}$  and  $\hat{c}$ . Additional local optimization over nearby values of  $u$  may also be performed to refine the estimate. A convenient method is to let



**Figure 13: Example of an ECDF with a Tail Estimate**

The function  $F_{Par}$  for a sample of size  $n = 300$  where  $Y \sim \text{Beta}(2, 2)$ . For  $x \geq u$ , the approximation (red) is given by a rescaled GPD CDF. The true distribution CDF (black) is shown for comparison.

$\delta = (y_n - y_1)/8$ , then perform some fixed number of optimization steps, in each of which:  $u$  is set to the best of  $\{u, u + \delta, u - \delta\}$ , and then  $\delta$  is halved. In our implementation, ten of these steps are applied after finding a reasonable estimate of  $u$ .

#### 7.4 Estimating the Parameter Distribution

With our GPD parameter estimates  $\hat{u}$ ,  $\hat{a}$ , and  $\hat{c}$ , we construct a modified ECDF of  $D$  that takes the shape of the usual step function when  $x < \hat{u}$ , and then switches to the smoother GPD CDF when  $x \geq \hat{u}$ :

$$F_{Par} = \begin{cases} 0 & \text{if } x < y_1, \\ i/n & \text{if } y_i \leq x < \min\{y_{i+1}, \hat{u}\}, \text{ and} \\ 1 - \frac{k(1 - \hat{F}_{\hat{u}}(x))}{n} & \text{if } x \geq \hat{u}, \end{cases} \quad (7.14)$$

where  $k$  is the number of negated sample points  $y_i$  such that  $y_i > \hat{u}$ . An example of  $F_{Par}$  for a  $\text{Beta}(2, 2)$  random variable is shown in Figure 13. Let  $R$  be a random variable with CDF given by  $F_{Par}(x)$ . With probability  $1 - k/n$ , a random number sampled from  $R$  will fall in the region defined by the step function, and with probability  $k/n$  it will fall in the region defined by the GPD tail approximation. The quantile function  $F_R^{\leftarrow}(p)$  is accordingly characterized in a piecewise manner. By the Inverse Transform Theorem, if  $N \sim U(0, 1)$ , then  $F_R^{\leftarrow}(N) \sim R$ .

With the upper tail attached to our model, we proceed with bootstrap. However, we now

use  $R$  instead of the empirical distribution of  $D$  as a model of  $Y$ . We draw 250 samples of  $n$  observations from  $R$ , and record the maximum of each sample as  $S_{max} = \{m_1, m_2, \dots, m_n\}$ . By the Fisher-Tippet-Gnedenko theorem, these sample maxima follow a GEV distribution for large  $n$ . The tail shape parameter  $\hat{c} = \hat{\xi}$  is an estimate of the shape parameter  $\xi$  of our GEV model. To estimate the remaining parameters  $\mu$  and  $\sigma$ , method of moments estimation is once again applied. Let  $H \sim GEV(\mu, \sigma, \xi)$ . Then for  $\xi < 0$ , we have

$$E(H) = \mu + \frac{\sigma(\Gamma(1 - \xi) - 1)}{\xi} \text{ and} \quad (7.15)$$

$$V(H) = \frac{\sigma^2 \Gamma(1 - 2\xi) - \Gamma(1 - \xi)^2}{\xi^2}. \quad (7.16)$$

This yields estimates for  $\mu$  and  $\sigma$  given by

$$\hat{\sigma} = -\hat{\xi} \sqrt{\frac{V(S_{max})}{\Gamma(1 - 2\hat{\xi}) - \Gamma(1 - \hat{\xi})^2}} \text{ and} \quad (7.17)$$

$$\hat{\mu} = E(S_{max}) - \frac{\hat{\sigma}(\Gamma(1 - \hat{\xi}) - 1)}{\hat{\xi}}. \quad (7.18)$$

Let  $Y_n$  be the distribution of the maximum of a sample of size  $n$  taken from  $Y$ . Define  $\hat{Y}_n \sim GEV(\hat{\mu}, \hat{\sigma}, \hat{\xi})$  as our model of  $Y_n$ . As a model of the error distribution  $Y_n - F_Y^{\leftarrow}(1 - 1/n)$ , we choose  $\hat{Y}_n - \hat{\mu}$ . This choice is based on the observation that  $\hat{Y}_n$  is centered on  $\hat{\mu}$ , while the distribution of the sample maximum  $Y_n$  is centered on  $F_Y^{\leftarrow}(1 - 1/n)$ . This gives us an estimate for the distribution of the extreme quantile  $F_Y^{\leftarrow}(1 - 1/n)$  of  $Y$ , given the sample information  $D$ , as

$$F_Y^{\leftarrow}(1 - 1/n) \sim y_n + \hat{\mu} - \hat{Y}_n. \quad (7.19)$$

Since  $Y = -X$ , we have  $F_Y^{\leftarrow}(1 - 1/n) = -\Omega_{1/n}$ . Then our parameter distribution estimate is

$$\Omega_{1/n} \sim \min(S) + \hat{Y}_n - \hat{\mu}. \quad (7.20)$$

This distribution estimate provides a CDF and a PDF to be used in discard comparisons, as described in Chapter 6.



## CHAPTER 8

### TEST FUNCTIONS

#### 8.1 Revisiting Ideal Bounding

In Chapter 2, we introduced branch and bound with an example that used “ideal” bounding in Figure 1. The problem is given by

$$\underset{0 \leq x \leq 3.2}{\text{minimize}} f(x) = 1 - \cos\left(\frac{\pi x^2}{2}\right). \quad (8.1)$$

We revisit this example with our inference-based bounding method in this section. Branches are once again closed intervals of  $\mathbb{R}$ , and branching is performed by bisection. Our method is applied using the  $1/n^{th}$  quantile as a guiding parameter, with confidence level 99% and sample size  $n = 200$ . Figure 14 demonstrates the progression of our optimization on (8.1).

A major difference in behavior between this and the ideal bounding behavior we described in Chapter 2 is that in our method, the second quarter is retained after the second bounding step. This is because based on the other branches’ parameter distributions, and on the steepness and height of the lower end of the distribution, a conservative estimate of the tail cannot confidently rule out a minimum in  $[0.8, 1.6]$  during that step.

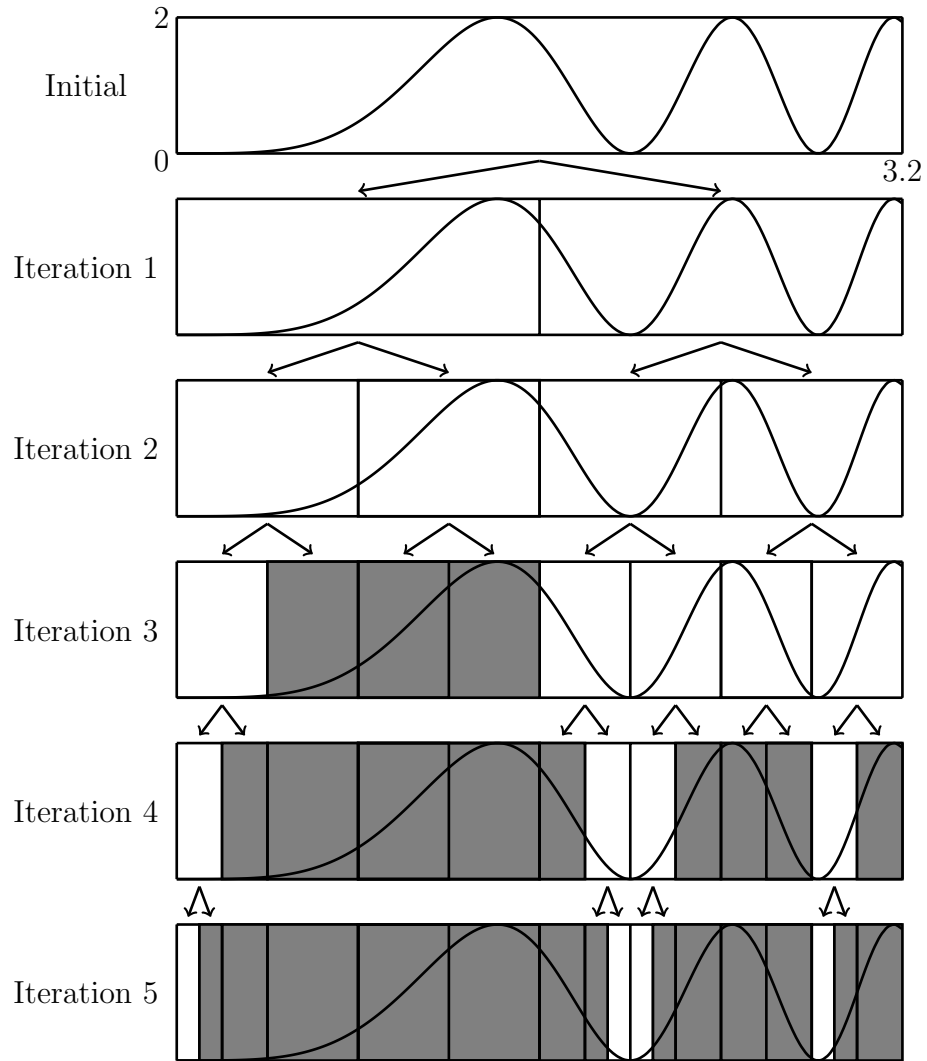
#### 8.2 The Wicked Comb

Many of our inferential bounding methods are intuitively less decisive when particular subsets of the branches have a high variance under the fitness function, and when the minima of different branches are close in value. To explore these conditions, a test function was devised where the number and closeness of local minima, as well as the variance within each subinterval, are controlled by user-specified parameters. The optimization problem is given by

$$\underset{|x| \leq 0.2}{\text{minimize}} f(x) = -a \cos\left(\frac{1}{|x| + \frac{1}{(2(b+0.25) \cdot \pi)}}\right) + cx^2 \quad (8.2)$$

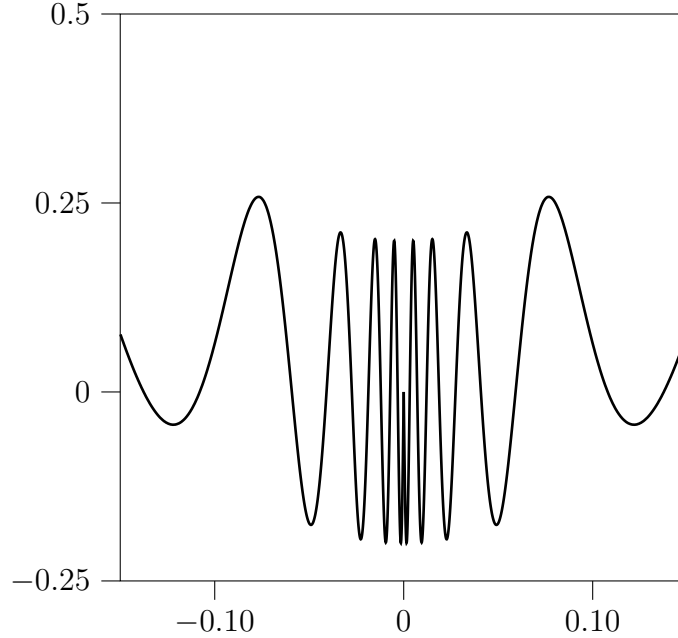
for some  $a, b, c \in \mathbb{R}^+$ .

The parameter  $b$  determines the number of local minima. For relatively small values of  $c$ ,



**Figure 14: Progression Diagram for the Optimization from Chapter 2**

In this visualization of the branches retained after each iteration of bounding on (8.1), discarded regions are displayed darkened.

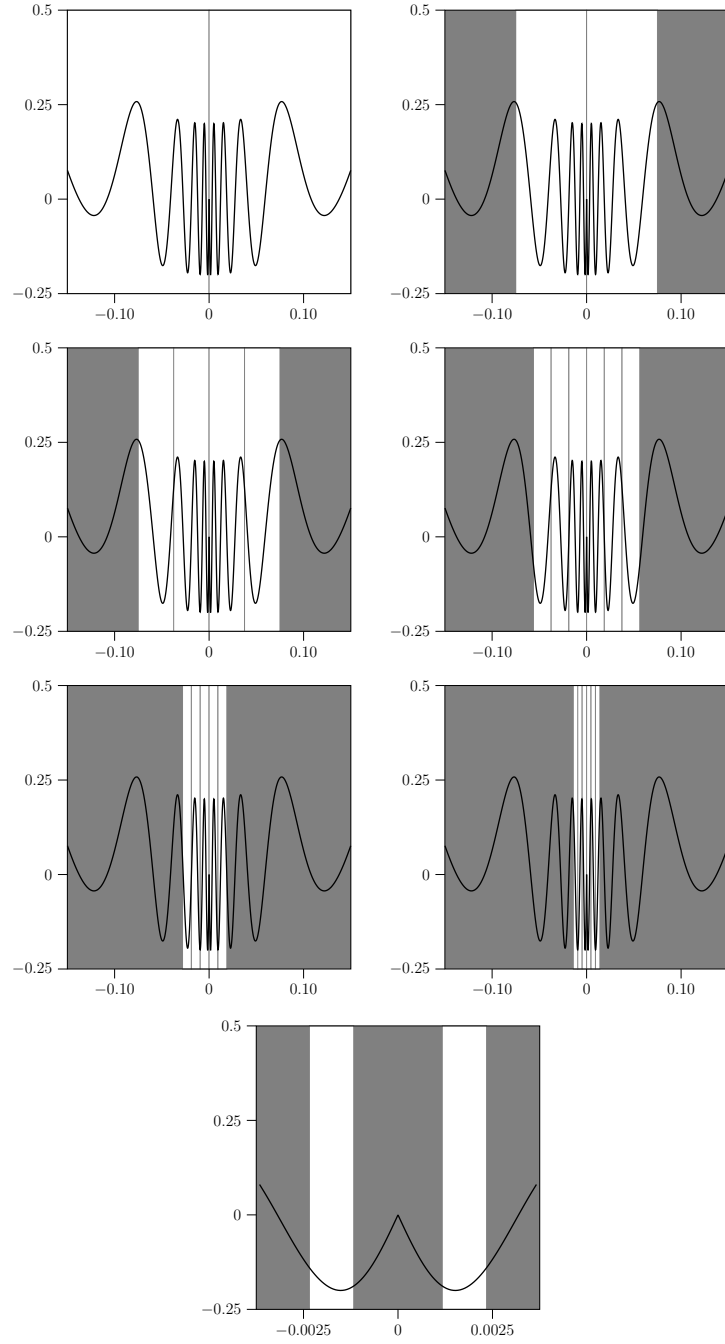


**Figure 15: Graph of the Wicked Comb Test Function**

The parameters for this particular version are given by  $a = 0.2$ ,  $b = 5$ , and  $c = 10$ .

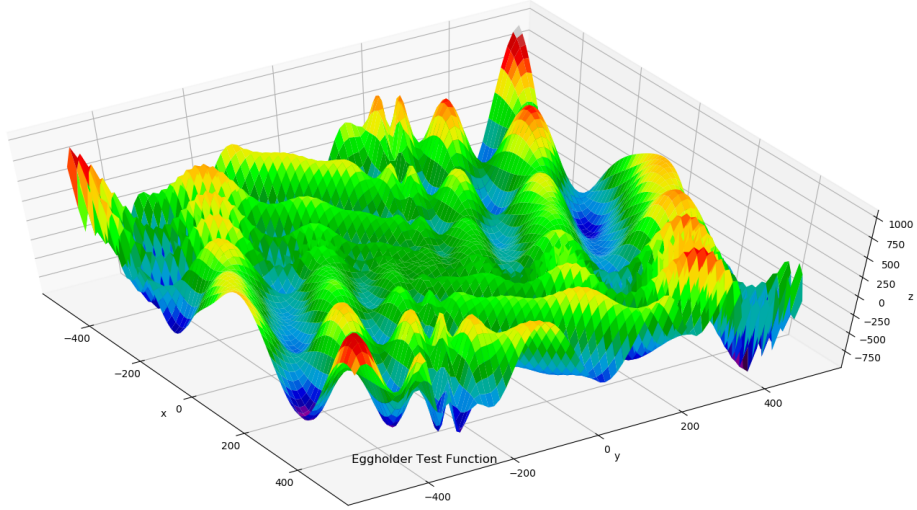
there will be  $2b$  many local minima, with  $b$  many of them on each side of  $x = 0$ . The  $cx^2$  term applies a gradual increase to the locations of the minima, making them closer under  $f$  as  $c \rightarrow 0$ , so that all local minima are global minima when  $c = 0$ . The parameter  $a$  rescales the function to determine the amount of variation in the values of  $f$ .

Figure 15 shows a Wicked Comb with  $a = 0.2$ ,  $b = 5$ , and  $c = 10$ , and Figure 16 demonstrates the convergence of the retained branches to the complete set of minima. The guiding parameter is the  $1/n^{th}$  quantile, the confidence level for discarding is 99%, and the sample size is  $n = 200$ . Branches are intervals of  $\mathbb{R}$ , and branching is performed via bisection. Note that despite having identical distributions of their function values, one of the two outer intervals in Step 5 were retained, while the other was discarded. This is possible because the sample from each interval is random, and if the tail estimate for one of these branches was longer, it may have been kept “just in case”. Meanwhile, the other interval may have yielded a clearer image of its distribution’s tail, and could therefore be discarded with greater confidence. Another possibility is that each had, for instance, 0.6% probability of being the best, so that only one could be discarded while remaining above 99% confidence of retaining the best, as described in Chapter 6.



**Figure 16: Progression Diagram for the Wicked Comb**

In this visualization of the branches retained after each bounding step, discarded regions are shown darkened. The last image skips ahead to after the 15<sup>th</sup> bounding step, and is zoomed to show the remaining two branches.



**Figure 17: Perspective View of the Eggholder Test Function**

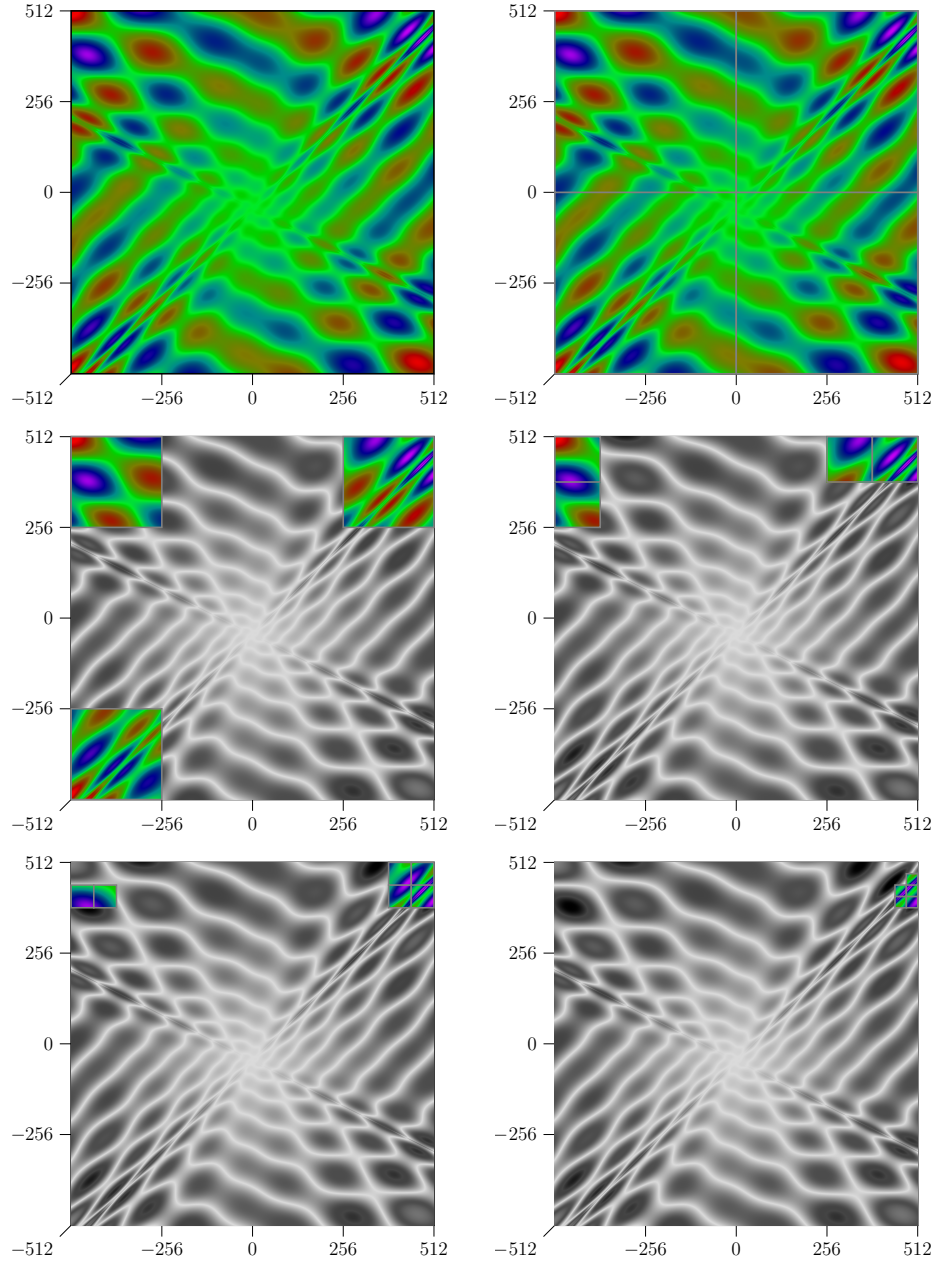
The global minimum may be observed near the right-most corner of the figure.

### 8.3 The Eggholder Test Function

The Eggholder is a 3D surface commonly used to test methods for box-constrained nonlinear optimization [11]. The optimization problem is given by

$$\underset{-512 \leq x, y \leq 512}{\text{minimize}} \quad f(x, y) = -(y + 47) \sin \sqrt{\left| \frac{x}{2} + y + 47 \right|} - x \sin \sqrt{|x - y - 47|}. \quad (8.3)$$

It has a single minimum at  $(512, 404.23, -959.64)$ . A perspective view of the surface is given by Figure 17. An example of the application of six steps of inference-driven branch and bound to the Eggholder test function is given by Figure 18. The guiding statistic is the  $1/n^{\text{th}}$  quantile, the confidence level used is 99%, and the sample size is  $n = 200$ . Branches are rectangular regions of  $\mathbb{R}^2$ , and branching is performed by quartering a given branch. The solution set contains the true minimum. Each of the regions retained overlap both high and low values, which encourages the tail estimates to be longer and the corresponding GEV distributions to have greater spread. As these intervals are steep, and also contain lower values that are competitive with the minimum, our method is less certain that they can be discarded.



**Figure 18: Progression Diagram for the Eggholder**

Retained branches after each step of the algorithm on the Eggholder test function. The top-left image is the initial state, the top-right is after bounding iteration 1, and so on. Retained branches are in color, and are emphasized by highlighted borders. Discarded regions are shown without saturation. Warmer colors correspond to higher elevations, with the lowest elevations shown in violet.

## CHAPTER 9

### APPLICATION TO AMBULANCE PLANNING

In seeking a natural progression from an existing plan to an optimized plan with a higher number of ambulances, the simplest and most robust choice would be to perform an exhaustive search of all possible progressions, choosing one that results in an optimal plan and, secondarily, performs well as it grows. Practically speaking, the number of such progressions is often too large to exhaust, as it grows exponentially with the number of ambulances to be added. In the specific case of Wayne County, there are ten possible locations to which we must consider adding an ambulance during each placement, and each ambulance can have either a part time or full time shift, so there are twenty options for each placement. Then the number of distinct progressions for adding  $n$  ambulances to an existing plan is  $20^n$ .

If the number of possible progressions is too large to exhaust, it is a natural next choice to try to perform local optimization - sometimes described as “hill-climbing” - where each new ambulance is added in the location and schedule that immediately improves the current plan the most. While this would produce a natural progression from the existing plan to one with more ambulances, this approach might be characterized as near-sighted; the placements that effect the greatest local improvement are not always the same as the placements that lead to the best overall improvement when the plan has reached the desired number of ambulances. For instance, if the geography of the region to be planned is circular, an early placement would likely be made at a central location with a full-time shift under local optimization. Meanwhile, the optimal solution after all placements have been made might require only circumferential placements. Furthermore, if among the choices for the next placement, another option offers nearly as much of an improvement as the best placement, it would still only use the best going forward without exploring that other option. Hill climbing also says effectively nothing about the relative importance of each placement to the performance of the final plan - indeed, only the last placement involves considering the performance of the final plan at all. Nonetheless, if the number of placements to be determined is small compared to the size of the existing plan, hill climbing is a sound approach for attaching a small number of additional ambulances to an

existing plan, and may be performed with different weightings on the objectives if greater variety is desired in the results.

These potential shortcomings of local optimization motivate our interest in a more robust approach. Ideally, we would add each ambulance with knowledge of how it will affect the optimality of the final plan, and in order of greatest to least importance to the final plan's performance. If multiple placements appear to lead to near-optimal outcomes, we should explore all of the most promising options rather than just one.

## **9.1 Data Used**

Two sources of data were used in this project: Wayne 911 and OpenStreetMap.org. Two years of de-identified ambulance response data were provided by Wayne 911, covering the years 2016 and 2017. A method for generating a rough estimate of the annual operating costs of a plan was also provided based on historical expenses. Additionally, a graph describing the roads of Wayne County and the surrounding region was obtained from OpenStreetMap.org and is used under the OpenStreetMap Geodata Licence.

### **9.1.1 Wayne 911 Data**

The following information was obtained from historical emergency response records for each of 12,487 responses spanning the years 2016 and 2017:

- Time Received - The time at which the call for this response was received.
- Time Enroute - The time at which an ambulance began traveling to the scene of the call.
- Time On-Scene - The time at which the ambulance arrived at the scene of the call.
- Time Departed - If the ambulance transported someone to the hospital, the time at which the ambulance departed the scene.
- Time Arrived - If the ambulance transported someone to the hospital, the time at which the ambulance arrived at the hospital.
- Time Completed - The time at which the ambulance is finished responding to the call and is available to respond again.



- Call Latitude, Longitude - The approximate position of the scene of the call, shifted to a nearby road for pathfinding purposes.

A rough cost estimate was also provided, which can be found in Section 9.2.

### 9.1.2 Open Street Map (OSM) Data

OpenStreetMap is a community-driven open-source service that provides map data for various applications [19]. We used this resource to obtain a road map of Wayne County and the surrounding region. Specifically, the data describe the roads in the region bounded by  $37.8409^{\circ}\text{N}$  to  $38.4944^{\circ}\text{N}$  in latitude, and from  $82.1853^{\circ}\text{W}$  to  $82.6804^{\circ}\text{W}$  in longitude. The data is in XML format, and is divided into two main parts: a list of nodes, which each have a unique 64-bit identifier as well as a latitude and longitude, and a list of “ways”, which contain a sequence of node identifiers that form a road as well as the name, zip code, county, and type of that road (eg residential, highway, etc.).

## 9.2 Goals and Metrics

Early on in our collaboration, we identified two essential properties of a plan that must be minimized in order for the plan to be considered optimal: response time and cost. The choice of how these properties are measured was informed by a consensus of opinion and convention.

Response time is measured by the arithmetic mean of the time between a call being received by the 911 center and the ambulance arriving on the scene on a per-response basis. That is, if two ambulances are dispatched to the same call, then two response times are recorded. This metric captures the actual travel time for the ambulance to arrive at the call, as well as any latency involved in dispatching ambulances and calling for mutual aid from additional stations and other counties. Other approaches using median response time or a trimmed mean were considered, and preliminary testing demonstrated similar results under optimization between mean and trimmed mean approaches. In the end, the arithmetic mean was chosen over the trimmed mean for its simplicity and because it takes every response time into account equally. A number of other summary statistics are also recorded, but they are not used directly as metrics of plan quality.

The cost of a plan is measured as the number of dollars which must be spent annually to

maintain it. This includes ambulance supplies, vehicle maintenance such as oil and tire changes, fuel, pay and benefits for employees such as drivers and technicians, station rent, and so forth. We worked with the 911 center to obtain a rough estimate of how these costs would likely scale given a fixed number of ambulances running on a given shift schedule. In our model, we use the formula

$$224F + 109P + 10S,$$

where  $F$  and  $P$  are the number of full and part-time ambulances operated, respectively, and  $S$  is the number of stations used by the plan. The result is interpreted in thousands of dollars annually. While more detailed cost computations would later be performed for specific plans, this formula gives us a “good enough” estimate for evaluating the cost of an arbitrary plan during optimization.

While estimating the cost of a theoretical plan using existing data is reasonably straightforward, estimating the mean response time of a plan is much harder. Wayne 911 provided us de-identified records of ambulance responses from the years 2016 and 2017, but those data only describe a single plan - the one that was used over those two years. Though there is some analysis that can be done with that information, it isn’t enough to estimate the likely performance of other plans. Instead, we developed a simulation model that lets us “re-live” those two years of 911 calls using an alternative plan, and calibrated this model using the real-world data. Though many assumptions and simplifications were necessary to enable it to run quickly, the simulation model managed to recreate many of the trends we observed in the real-world data and remains our best method of estimating the mean response time of a plan without trying it out in real life. More information on the simulation model can be found in Section 9.3.

Intuitively, response time and annual cost are dependent objectives - improvement in one often comes at some expense in the other. At one extreme, an empty plan will cost nothing and have an infinite mean response time. At the other, a plan with 100 ambulances active at all hours parked at every station will have a very low mean response time and exceedingly high costs. The objectives of minimizing cost and response time are in direct competition with one another, which is precisely the kind of relationship that merits phrasing the problem in the terms of

multi-objective optimization. More information on multi-objective optimization and how it relates to the ambulance planning problem can be found in Section 2.1.2.

The result of this consideration is a metric that describes the “fitness” of a plan for the purpose of comparing two plans. The fitness of an ambulance distribution plan  $d$  is given by the product

$$f(d) = R(d)C(d),$$

where  $R$  is a function that estimates the mean response time of  $d$  and  $C$  is a function that estimates the annual cost of  $d$ . Intuitively, the smaller a plan’s fitness score, the more cost-efficient the plan will be. This particular choice of fitness function was motivated by a desire to have a proportional increase in each of the objectives weighted equally. For instance, a 10% improvement in the cost of a plan results in exactly the same change to the plan’s fitness as a 10% improvement in mean response time. Each would result in a 10% improvement in the fitness score.

### 9.3 EMS Simulation Model

The purpose of our simulation model is to use existing data that describe how the ambulance services behaved in Wayne County during 2016 and 2017 to estimate how alternative placements of ambulances would have performed over the same time period. First, it uses historical response data and a road map to compute a matrix of estimated travel times to the location of each call from each station. Then, it runs an event-driven re-enactment of the calls that came in over the two years for which we have data, and simulates the behavior of ambulances in response to these calls. During this, the simulation keeps track of ambulance availability and measures response times, mutual aid coverage, the amount of time each ambulance is in use, and so on. A complete list of output data is provided in Section 9.3.3. The simulation can be performed with any arrangement of ambulances, and each ambulance can be independently scheduled. This allows us to examine alternative station/parking locations and shift schedules in addition to adding ambulances to existing locations.

### 9.3.1 Assumptions and Simplifications

The behavior of ambulances is affected by many processes and factors that are not always simple to predict. In order to model this behavior with a simulation, many simplifications were made. This subsection lists the assumptions and simplifications that were used in the development of the simulation model.

- **All calls are equally important.** In the simulation, calls are responded to in the order that they are received by 911. All calls requiring an ambulance response are assumed to be equally urgent.
- **Recent data can describe present and future behavior.** The years 2016 and 2017 are assumed to be representative in terms of the frequency and location of calls. The potential for a rapid change in the volume of incoming calls, or in the geographic regions from which calls are most common, is not considered by the model.
- **Dispatching an ambulance is instantaneous.** It is assumed that if there are ambulances available when a call begins, an ambulance will be dispatched immediately to respond. There is no simulation of the delay involved in finding the most appropriate ambulance to respond to the call and sending it out.
- **The average speed of an ambulance is constant.** Ambulances always travel at the same average speed in the simulation. This speed was calibrated to 24 miles per hour by choosing the speed that enabled the simulation to most closely match the observed response times of real ambulances under the configuration used in 2016 and 2017. Vehicle acceleration and traffic conditions are not simulated by the model.
- **Stations do not proactively respond to the needs of other stations.** In the real world, if a station sends out all of its ambulances, and another station nearby has an available ambulance, it may send that ambulance to halfway between the two stations so that it can readily respond to incoming calls in either station's service region. This potential behavior is not modeled by the simulation.

- **The curvature of the earth can be ignored.** The simulation models the roads of Wayne county as a weighted graph on a flat surface.
- **All roads are equally navigable.** In reality, many roads in the county are sharply curved and some are not paved. While such roads might be navigated more slowly than others in the real world, we assume all roads can be traveled at the same speed for the simulation model.
- **Patients always choose the nearest hospital.** All ambulances transporting someone to a hospital are assumed to go to the nearest hospital. In the real world, the person being transported can dictate the hospital to which they want to be delivered, but as we do not have this information for most of the calls, it is assumed for the simulation that every person being transported will choose the nearest hospital.
- **Each ambulance responds from its station.** In a previous version of the simulation, we tracked each ambulance's position on the road graph as it traveled. With this information, a nearby ambulance could respond to an incoming call on its way back to the station from another call. This diversion mid-path was observed to occur very rarely, however - usually either the ambulance had a call to respond to immediately when it became available, or made it back to the station before the next call came in. The simulation was sped up significantly by simplifying this behavior in a later version, which we used in this project. In this version, an ambulance always responds along a path from its station to the call location.

### 9.3.2 Implementation

The simulation model, along with most of the rest of the project, was written in C# as a console application targeting the .NET Core platform for compatibility with both Windows and Linux machines. This was chosen because some of the optimization methods we used involved distributing the processing load over the nodes of a compute cluster, which is a Linux environment, while the application itself was developed on Windows machines. While the overall procedure followed by the simulation is outlined here, it is unavoidable that some nuances are omitted for brevity's sake. Additional information can be found in the source code, which is

available on GitHub [15] and contains XML documentation.

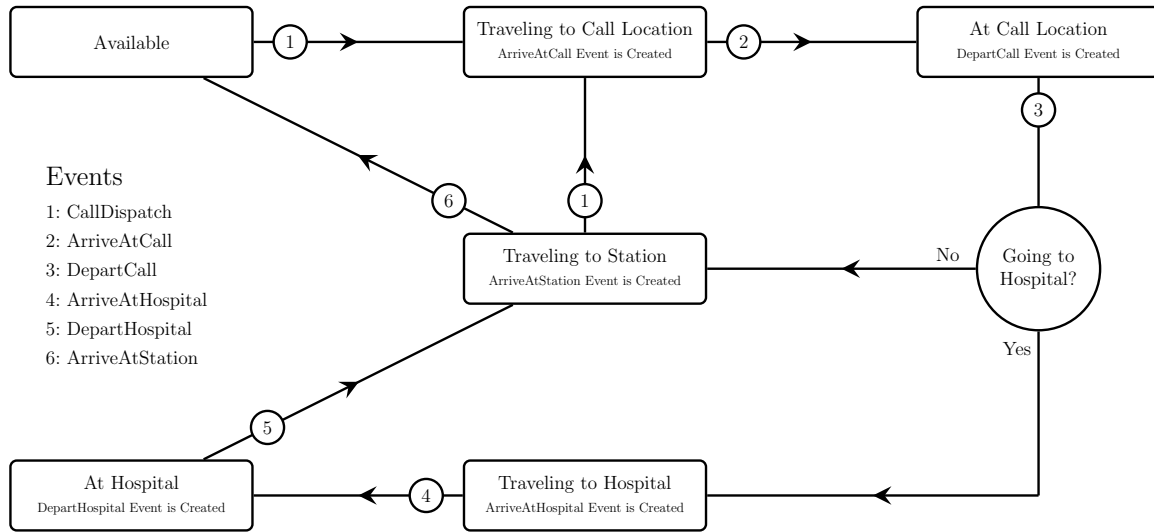
The simulation begins by generating a table that describes the travel time from each call location to each station and hospital. This table can alternatively be imported if it has been generated for a previous run with the same data. Each travel time is computed by first determining the distance to be traveled, then dividing by the average travel speed of an ambulance, which is assumed to be constant. The distance to be traveled is computed by finding the nearest nodes to the point of departure and destination, respectively, and pathfinding along the road graph to obtain a sequence of nodes that connect them. Pathfinding is performed using the A\* algorithm [8]. A downscaled  $L^1$  norm is used for the pathfinding heuristic, given by

$$h(\mathbf{a}, \mathbf{b}) = \left| \frac{a_x + a_y - b_x - b_y}{\sqrt{2}} \right|,$$

where  $\mathbf{a}$  and  $\mathbf{b}$  are nodes on the road graph with positions given by  $(a_x, a_y)$  and  $(b_x, b_y)$ , respectively. The  $1/\sqrt{2}$  factor is included to prevent overestimation, which may result in sub-optimal pathfinding if it occurs.

Once the table of travel times has been obtained, the simulation’s current time is set to midnight on January 1, 2016, and a list of events that will occur during the simulation is populated. Every call corresponds to an event that is scheduled to occur at the time the call was received, at which point an ambulance response is attempted for that call. When an ambulance is dispatched to respond to a call, its state is changed from “available” to “traveling to call,” and an arrival event is added to the event list at the current time plus the travel time from that ambulance’s station to the call. When this arrival event occurs, the ambulance’s state transitions to “at call,” a departure event is added to the list, and so on. The behavior of ambulances in the simulation is described by the finite state machine outlined in Figure 19.

During each event, appropriate records for each ambulance and station are updated. The simulation progresses by advancing time to the next scheduled event and processing that event, repeating this until there are no more events in the list.



**Figure 19: Flowchart of Simulated Ambulance Behavior**

A flowchart of the finite state machine that governs ambulance behavior in the simulation model.

### 9.3.3 Generated Data

When the simulation model is run, it records data that can be used to gain insight into the performance of various elements of the simulated ambulance distribution. The information that is recorded in each run is summarized here.

- **Call Response Times.** Response time is measured as the duration between when a call is received by 911 and when an ambulance arrives at the call's location. The entire set of response times can be exported for analysis, but usually only a summary is recorded. The summary includes the arithmetic mean, inter-quartile mean, and percentiles at increments of 10%. Additionally, the mean response time for each station is recorded.
- **Call Counts.** The total number of ambulance responses that are sent out from each station is recorded.
- **Queued Calls.** When a call is received by 911 but no ambulances are currently available to respond, it is momentarily enqueued until the next ambulance is available. The number of times this happens is recorded.
- **Mutual Aid.** When a call is received and the closest station does not have any ambulances

available to respond, another station will cover the call. This process is referred to as mutual aid, and the number of times each station covers each of the other stations is tabulated.

## 9.4 Optimization

In 2018, an exhaustive search optimization for the ambulance planning problem was run on the Big Green Cluster - Marshall University's high-performance computing solution. A heatmap of the solution space for 10 ambulances is shown in Figure 20. We compare our branch and bound results against the solutions found by the exhaustive search as a benchmark.

### 9.4.1 Branching

A solution in the context of the ambulance planning problem is a matrix  $P \in \mathbb{N}^{10 \times 2}$ , where the first column lists how many ambulances are allocated to the  $i^{th}$  station on a full-time schedule, and the second column describes in the same manner the placement of ambulances with part time schedules.

We consider two approaches to defining **Branch()** for the ambulance planning problem.

**Method A.** Define

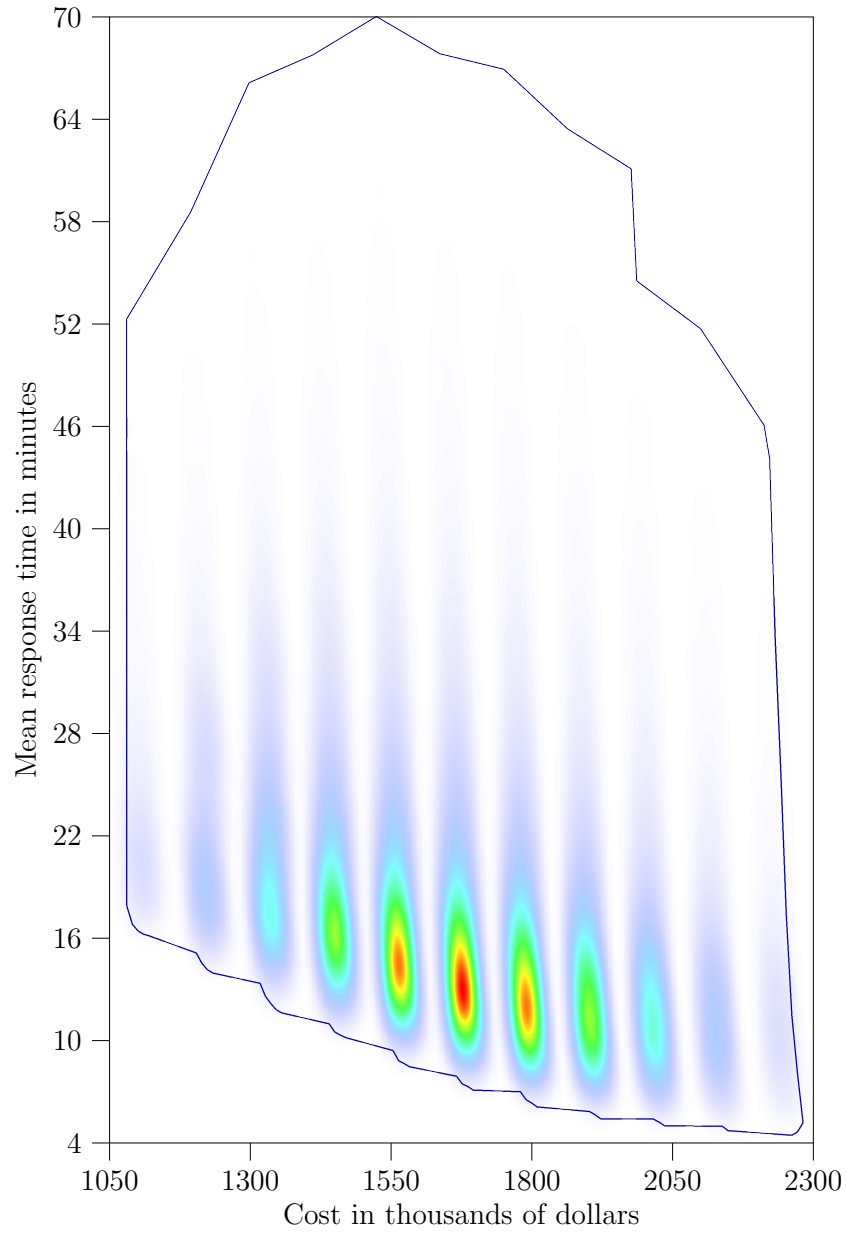
$$\text{Branch}(P) = \{P + 1_{i,j} \mid 1 \leq i \leq 10, j = 1, 2\}, \quad (9.1)$$

where  $1_{i,j}$  is a  $10 \times 2$  matrix with all entries zero except for a 1 at position  $i, j$ . This corresponds to looking at every way that it is possible to add one ambulance to the plan. This approach yields 20 new branches from each retained branch after bounding, though these new branches are not guaranteed to be unique, as there will be some overlap if two or more branches are retained.

Optimization under this manner of branching can be characterized as attempting to determine the appropriate position and schedule for the next ambulance to be added to the plan, with the goal of later achieving an optimal plan for a larger number of ambulances. This method may produce constructive solutions, but as there are some overlapping elements between the branches, it is not always possible to uniquely determine the path that was taken to arrive at a given result.

**Method B.** Suppose there are  $c$  stations, and choose a permutation  $(s_1, s_2, \dots, s_{2c})$  of station-shift pairs. Four ambulances is a practical upper bound on the number of ambulances that could be allocated to a single station-shift pair. We begin, as before, with an empty or existing





**Figure 20: Heatmap of the Solution Space for Plans with 10 Ambulances**

A boundary line (blue) is drawn along the outermost solutions. The Pareto set is located within the wavy curve that forms the lower-left portion of the boundary line.

plan. In the  $i^{th}$  branching step, we consider adding 0, 1, 2, 3, or 4 ambulances to the  $i^{th}$  station-shift in the permutation, within the maximum number of total ambulances. This method may be characterized as first deciding how many ambulances to place in  $s_1$ , then deciding the same for  $s_2$ , and so on. The results are not constructive, as the order of ambulance placement is determined by the permutation and not by their importance to the fitness of the plans in the resulting branches. This branching method produces a partition of the search space, and fewer branches are generated as the result of each branch retained during bounding, so it holds promise for efficiently finding non-constructive solutions.

## 9.5 Results

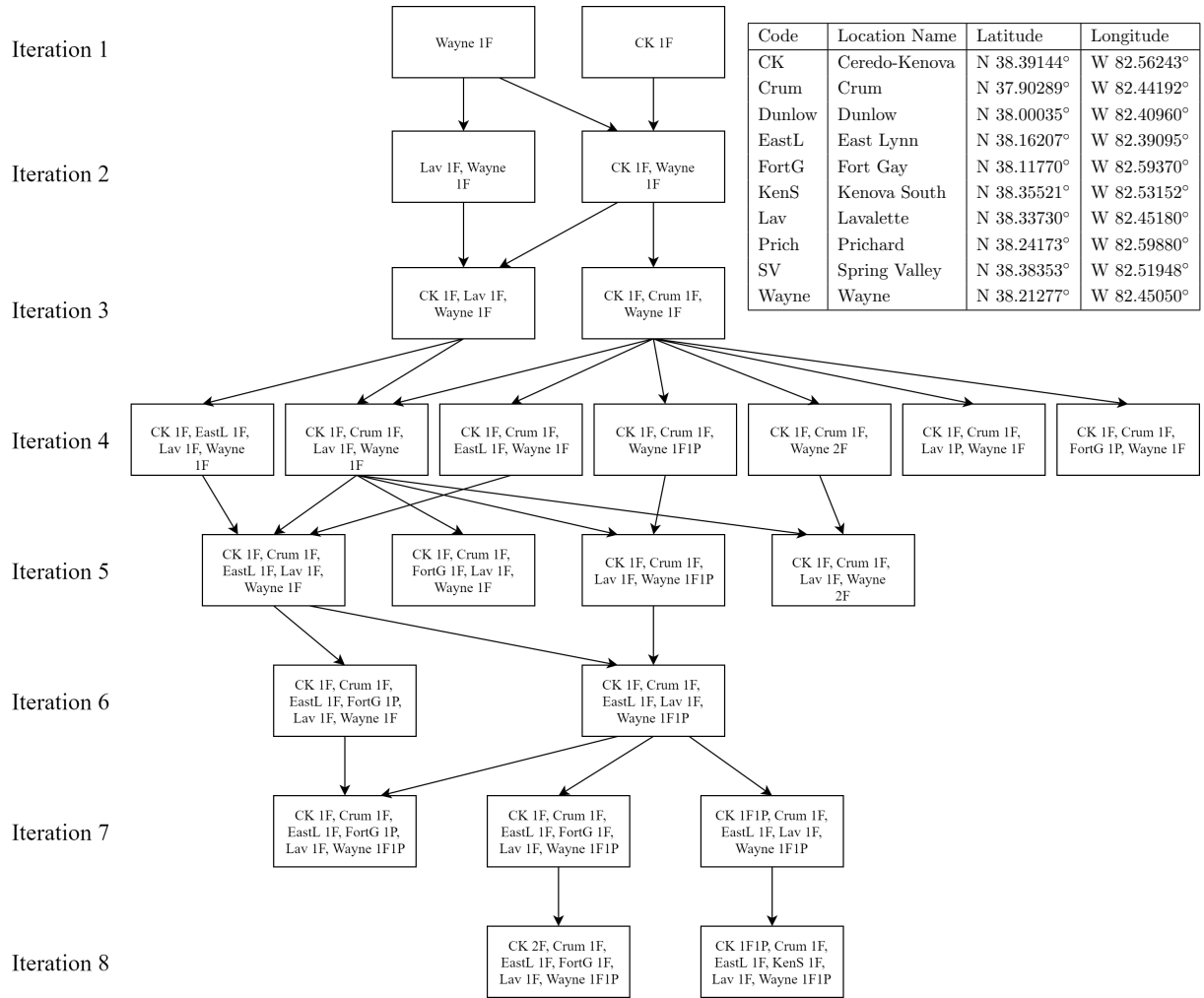
The results of running a branch and bound optimization to place 10 ambulances are detailed in Figure 21. These were obtained with a sample size of 1000, a confidence level of 95%, and a guiding parameter given by the mean of the fitnesses that are less than the 10<sup>th</sup> percentile. Branching was performed using Method A. The optimization ends after placing 8 ambulances, and there are fewer than 800 solutions in the union of the resulting branches. Among these solutions are multiple Pareto optimal solutions identified in the exhaustive search.

## 9.6 Future Work

The inference-driven branch and bound optimization method we have developed and implemented in this work is at a proof-of-concept level of revision. We have demonstrated that this method can be effective for solving optimization problems of both real and academic origin. However, there are many directions through which future research may improve, or at least better describe, the limitations of the method. A number of these have been mentioned in previous chapters, and some salient topics for future study are listed here.

- **Improved Tail Estimation and Model Fitting for Extreme Quantiles**

Our present implementation uses method of moments estimation and a number of simplifications, including assuming that the location parameter of a GEV error distribution is always effectively zero. Better tail estimation and model fitting will significantly improve the reliability of the  $1/n^{th}$  quantile and similar extreme quantile guiding parameters.



**Figure 21: Progression Diagram for an Application of Branch and Bound**

The progression diagram for an application of inference-driven branch and bound to the emergency vehicle planning problem. The branches retained after each iteration are listed as rectangles, and contain a list of their allocated ambulances. Each entry can be read as a station identifier, followed by  $aFbP$  to denote  $a$  full-time and  $b$  part-time ambulances at that station.

- **Faster, Simpler, and More Reliable Integration of the Complement Discard Probability Integral**

The quadrature methods we considered trade between simplicity, efficiency, and reliability. Extensive additional testing and broader consideration of alternative methods may yield more reliable and efficient computational methods for this purpose.

- **Automatic Branching**

Presently, the user must specify a branching method for the solution space and subsequent branches. While this permits the selection of branching methods that suit the user's preferences, an automated, data-driven approach to branching may be more effective for optimization.

- **Adaptive Guiding Parameters and Statistics**

Additional guiding parameters and statistics may permit more effective convergence to optimal solutions. In particular, estimates that adapt to make more effective decisions during later iterations may improve the results of the optimization.

- **Automatic and Adaptive Choice of Sample Size**

The sample size is presently left to the user to determine, and does not change during the optimization. Adaptive sample sizes may yield computational savings by judging that an additional branch may be discarded with a modest augmentation of the branch samples, or by determining that the same conclusions could have been achieved with a smaller sample.

- **Analysis of Error Propagation**

At present, each bounding operation uses a fixed confidence level for discarding branches, without regard for the possibility that the optimal value of the guiding parameter may belong to a branch that was discarded in previous iterations. A more careful analysis of how confident we can be that the optimal value of the guiding parameter is being retained at each step may illuminate useful improvements to the method.

- **Analysis of Robustness to Errors in the Objective Function for each Guiding Parameter**

Most of our work has assumed that the objective function  $f$  is correct. Our emergency vehicle planning optimization is about minimizing  $f = RC$ , which is not necessarily using the real world annual cost or mean response time, and the results of the optimization may perform poorly in real applications if the models we use to measure those objectives are inaccurate. Another reason we may not be able to fully trust the objective function could be if it contains random measurement errors, which occur in many applications [24]. Some guiding parameters may be more robust than others under these conditions.

## REFERENCES

- [1] Johannes M. Bader, *Hypervolume-Based Search for Multiobjective Optimization*, (2009), 1–17.
- [2] A. A. Balkema and L. de Haan, *Residual Life Time at Great Age*, Ann. Probab. **2** (1974), no. 5, 792–804.
- [3] David Blackman and Sebastiano Vigna, *Scrambled Linear Pseudorandom Number Generators*, 2018.
- [4] I. Bogaert, *Iteration-Free Computation of Gauss–Legendre Quadrature Nodes and Weights*, SIAM Journal on Scientific Computing **36** (2014), no. 3, A1008–A1026.
- [5] John Burkardt, *Gauss-Hermite Quadrature Rules*, [https://people.sc.fsu.edu/~jburkardt/cpp\\_src/hermite\\_rule/hermite\\_rule.html](https://people.sc.fsu.edu/~jburkardt/cpp_src/hermite_rule/hermite_rule.html), 2014.
- [6] C. W. Clenshaw and A. R. Curtis, *A Method for Numerical Integration on an Automatic Computer*, Numer. Math. **2** (1960), no. 1, 197–205.
- [7] Laurens de Haan and Ana Ferreira, *Extreme Value Theory: An Introduction*, 1st edition. ed., Springer, 2010.
- [8] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, Systems Science and Cybernetics, IEEE Transactions on **4** (1968), 100 – 107.
- [9] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability, Taylor & Francis, 1994.
- [10] James Pickands III, *Statistical Inference Using Extreme Order Statistics*, Ann. Statist. **3** (1975), no. 1, 119–131.
- [11] Momin Jamil and Xin She Yang, *A Literature Survey of Benchmark Functions for Global Optimisation Problems*, International Journal of Mathematical Modelling and Numerical Optimisation **4** (2013), no. 2, 150.
- [12] Samuel Kotz and Saralees Nadarajah, *Extreme Value Distributions.*, Imperial College Press, 2000.
- [13] A. H. Land and A. G. Doig, *An Automatic Method of Solving Discrete Programming Problems*, Econometrica **28** (1960), no. 3, 497–520.
- [14] Hubert W. Lilliefors, *On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown*, 1967.
- [15] Kevin C. McDaniel, *EMS Optimization and Distributed Search Application*, <https://github.com/mcdaniel55/EMSOptimizationV2>, 2019.
- [16] ———, *Statistical Inference Driven Branch and Bound Implementation*, <https://github.com/mcdaniel55/Thesis>, 2019.

- [17] Kurt Mehlhorn and Peter Sanders, *Algorithms and Data Structures: The Basic Toolbox*, 1 ed., Springer Publishing Company, Incorporated, 2008.
- [18] National Academy of Sciences and National Research Council, *Accidental Death and Disability: The Neglected Disease of Modern Society*, The National Academies Press, Washington, DC, 1966.
- [19] OpenStreetMap contributors, *Planet dump* retrieved from <https://planet.osm.org>, <https://www.openstreetmap.org>, 2017.
- [20] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3 ed., Cambridge University Press, New York, NY, USA, 2007.
- [21] Timothy Sauer, *Numerical Analysis*, 2nd ed., Addison-Wesley Publishing Company, USA, 2011.
- [22] Lloyd N. Trefethen, *Is Gauss Quadrature Better than Clenshaw-Curtis?*, SIAM Review **50** (2008), no. 1, 67–87.
- [23] Lloyd N Trefethen and JAC Weideman, *The Exponentially Convergent Trapezoidal Rule*, SIAM Review **56** (2014), no. 3, 385–458.
- [24] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer, *Mathematical Statistics with Applications*, seventh ed., Duxbury Advanced Series, 2002.
- [25] Jrg Waldvogel, *Fast Construction of the Fejer and Clenshaw Curtis Quadrature Rules*, BIT Numerical Mathematics **46** (2006), 195–202.

**APPENDIX A**  
**LETTER FROM INSTITUTIONAL RESEARCH BOARD**



Office of Research Integrity

June 20, 2019

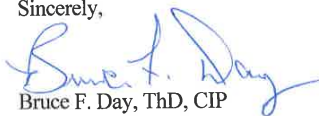
Kevin McDaniel  
415 8<sup>th</sup> Ave.  
Huntington, WV 25701

Dear Mr. McDaniel:

This letter is in response to the submitted thesis abstract entitled "*An Inference-Driven Branch and Bound Optimization Strategy for Planning Ambulance Services.*" After assessing the abstract, it has been deemed not to be human subject research and therefore exempt from oversight of the Marshall University Institutional Review Board (IRB). The Code of Federal Regulations (45CFR46) has set forth the criteria utilized in making this determination. Since the information in this study does not involve human subjects as defined in the above referenced instruction, it is not considered human subject research. If there are any changes to the abstract you provided then you would need to resubmit that information to the Office of Research Integrity for review and a determination.

I appreciate your willingness to submit the abstract for determination. Please feel free to contact the Office of Research Integrity if you have any questions regarding future protocols that may require IRB review.

Sincerely,



Bruce F. Day, ThD, CIP  
Director

**WE ARE... MARSHALL.**

One John Marshall Drive • Huntington, West Virginia 25755 • Tel 304/696-4303  
A State University of West Virginia • An Affirmative Action/Equal Opportunity Employer