Global and Robust Optimization for Engineering Design

by

Berk Öztürk

S.M., Massachusetts Institute of Technology (2018)
B.S., Massachusetts Institute of Technology (2016)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Berk Öztürk
Department of Aeronautics and Astronautics
January 16, 2022

Certified by . . . .
Dimitris Bertsimas
Boeing Professor of Operations Research, Sloan School of Management
Thesis Supervisor

Certified by . . . .
Mark Drela
Terry J. Kohler Professor, Department of Aeronautics and Astronautics
Committee Member

Certified by . . . .
Robert Haimes
Principal Research Engineer, Department of Aeronautics and Astronautics
Committee Member

Accepted by . . . .
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Global and Robust Optimization for Engineering Design

by

Berk Öztürk

Abstract

There is a need to adapt and improve conceptual design methods through better optimization, in order to address the challenge of designing future engineered systems. Aerospace design problems are tightly-coupled optimization problems, and require all-at-once solution methods for design consensus and global optimality. Although the literature on design optimization has been growing, it has generally focused on the use of gradient-based and heuristic methods, which are limited to local and low-dimensional optimization respectively. There are significant benefits to leveraging structured mathematical optimization instead. Mathematical optimization provides guarantees of solution quality, and is fast, scalable, and compatible with using physics-based models in design. More importantly perhaps, there has been a wave of research in optimization and machine learning that provides new opportunities to improve the engineering design process. This thesis capitalizes on two such opportunities.

The first opportunity is to enable efficient all-at-once optimization over constraints and objectives that use arbitrary mathematical primitives. This work proposes a constraint sampling and learning approach for global optimization, leveraging developments in machine learning and mixed-integer optimization. More specifically, the feasible space of intractable constraints is sampled using existing and novel design of experiments methods, and learned using optimal classification trees with hyperplanes (OCT-Hs). OCT-Hs describe union-of-polyhedra approximations of intractable constraints, which are solved efficiently using commercial solvers to find near-feasible and near-optimal solutions to the global optimization problem. The constraints are then checked and the solution is repaired using projected gradient methods, ensuring feasibility and local optimality. The method is first tested on synthetic examples, where it finds the global optima for 9 out of 11 benchmarks, and high-performing solutions otherwise. Then it is applied to two real-world problems from the aerospace literature, and especially to a satellite on-orbit servicing problem that cannot be addressed via other global optimization methods. These applications demonstrate that decision tree driven optimization provides efficient, practical and optimal solutions to difficult global optimization problems present in aerospace design as well as other domains, regardless of the form of the underlying constraints.

The second opportunity is to optimize designs affected by parametric uncertainty in a tractable and deterministic manner, while providing guarantees of constraint satisfaction. Inspired by the wealth of literature on robust optimization, and specifically on robust geometric programming, this thesis proposes and implements robust signomial programming to solve engineering design problems under uncertainty. The methods are tested on a conceptual aircraft design problem, demonstrating that robust signomial programs are sufficiently general to address engineering design problems, solved efficiently by commercial solvers, and result in designs that protect deterministically against uncertain parameter outcomes from predefined sets. In addition, robust designs are found to be less conservative than designs with margins; robust aircraft demonstrate 9% better average performance than aircraft designed with margins over the same scenarios, while providing guarantees of constraint feasibility.

In anticipation of future aerospace design problems becoming increasingly coupled, complex and risky, this thesis provides a new perspective for dealing with design challenges using structured mathematical optimization. The proposed methods inject mathematical rigor into engineering design methods while keeping practical concerns for conceptual design in focus.

Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Professor of Operations Research, Sloan School of Management

Committee Member: Mark Drela
Title: Terry J. Kohler Professor, Department of Aeronautics and Astronautics

Committee Member: Robert Haimes
Title: Principal Research Engineer, Department of Aeronautics and Astronautics

Acknowledgements

This thesis has had a particularly windy path to success, and it would not have been possible if it were not for the people who believed in me, gave me opportunities, and supported me unconditionally.

I would like to thank my first advisor Woody Hoburg, for giving me an opportunity in his lab and putting me on a course of academic self-actualization, for lack of a better description. I still look back fondly at our group meetings, where we would discuss research over mouthfuls of pizza. It was there that I developed a passion for using optimization to improve conceptual design, and started asking some of the questions that I would try to address in this thesis.

After his departure from MIT, I was supervised by Bob Haimes and Prof. Mark Drela. As I tried to find my own research path, my second and third years in graduate school were not pretty. Mark and Bob understood that it was not for lack of trying; they supported me despite the challenges and setbacks, gave me latitude to explore different areas of inquiry, and always gave honest feedback that pushed me to improve. Their research, their conceptual design philosophy and our discussions have inspired me greatly. I want to thank Bob doubly for being an amazing mentor; he is one the most genuine and caring people I know, and I leant on him for support more times than I can count. Without Bob, I would have pulled the ripcord on graduate school a long time ago.

Prof. Dimitris Bertsimas was an invaluable final addition to my committee. Even though Dimitris was my research advisor only during the last two years, Dimitris' influence is felt throughout this thesis. His classes and research on robust optimization and machine learning provided the foundations and major inspirations for my research. When I approached Dimitris with a proposal for global optimization in Fall 2019, he graciously took me in, and somehow squeezed more meetings into his busy calendar. Dimitris helped me direct my creative energy, and provided critical feedback guiding me to success. He cared deeply in helping me succeed; thank you for your mentorship and support.

I would like to thank my thesis readers Dr. Jack Dunn and Prof. Oli de Weck for their feedback and guidance through the many phases of my PhD career; Prof. Wes Harris for his leadership as a housemaster and his presence in my committee deliberations; Ali for being a great friend, collaborator and RGP savant; Dave Robertson and Todd Billings for always helping me out with personal projects and for the conversations; my 15.095 students for motivating me to be the best TA I could be; and finally many other collaborators, labmates and friends: the 16.82 JHO teams, Riley, Adam, the ACDL, the ORC, MakerWorkshop, the Convex Engineering Group, the Aircraft Design Club. My sincere apologies for any omissions.

Four groups deserve special mention. Many thanks to the MIT Cycling Club for taking me in and enabling me. I had no idea that I would meet some of my best friends in graduate school and beyond on two wheels; you could not find a more amazing community of caring and passionate people.

Thanks to the students of Desmond, New House, for sharing their MIT experience with me and Elise as their GRAs. I am looking forward to what you will accomplish in your personal and academic lives beyond MIT.

Thanks to the New House GRAs for the amazing work they do supporting students, and for their friendship, which brightened up even the darkest pandemic days.

My friends in Airbus (not the company), you are one of the few constants in my life. I will never take your friendship for granted.

Thanks to the many other friends without whom I couldn't have made it. Unfortunately, that net is cast very wide, so you will remain unnamed lest I forget someone.

Now to family. It's hard to thank my parents Erdinç and Sevgi Öztürk without trivializing their contributions. Thank you for everything.

Thanks to my second family Kara, Stu, Blaine and Will, for accepting me as their own.

Deniz, it has been a privilege to watch you grow from a small boy to a young man during the last five years. I am proud of you and I hope you are proud of me too.

Elise, you are the sunshine of my life. You make me a better person, and I hope that the stars will align and we will be reunited soon.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conceptual design is indispensable in aerospace engineering, as we push the envelope of what is possible with novel vehicle configurations. Aerospace engineers are ever approaching the limits of the Second Law of Thermodynamics, as they strive for the utopia of faster, cleaner, more efficient and less noisy. This thesis is motivated by the need to adapt and improve conceptual engineering design methods through better optimization, to address the challenge of designing future systems. We assert that conceptual design tools of the future need to achieve two objectives. The first is to optimize efficiently over a combination of constraints, models and data without placing restrictions on their mathematical forms. The second is to capture the effects of uncertainty on design decisions. This thesis proposes tractable and practical methods that aim to achieve these objectives.

1.1 Challenges and trends in aerospace design

Aerospace design is a particularly complex systems engineering problem. It is multidisciplinary, requiring knowledge of physics in many technical disciplines, but also in the social sciences such as economics and sociology. The interaction of the disciplines is tight, and the resulting complexity is combinatorial in the number of domains. Aerospace projects are often expensive and long term, and therefore have high risk and exposure to uncertainty. There is also little legacy design experience, since very few aerospace products have been developed and in very limited configurations. Since the aerospace sector is competitive and secretive, most existing design knowledge is proprietary.

Due to a combination of challenges, the industry has not been able to keep on top of schedule and cost overruns while developing increasingly complex systems since the 1950's. This is especially clear in military aerospace in Figure 1-1, where 'clean sheet' designs such as the B-2 and the V-22 have seen precipitous linear growth in development time since the mid-1970's. Even in the context of commercial aircraft, where design progress has been incremental and there is significant economic pull for fast design cycles, there has been significant increase in number of years to operational capability. This is in contrast with the automotive industry which has been improving its design process efficiency dramatically.

These trends are dramatized by the fact that our computational capabilities have increased substantially over the same time period. As shown in Figure 1-2, the number of

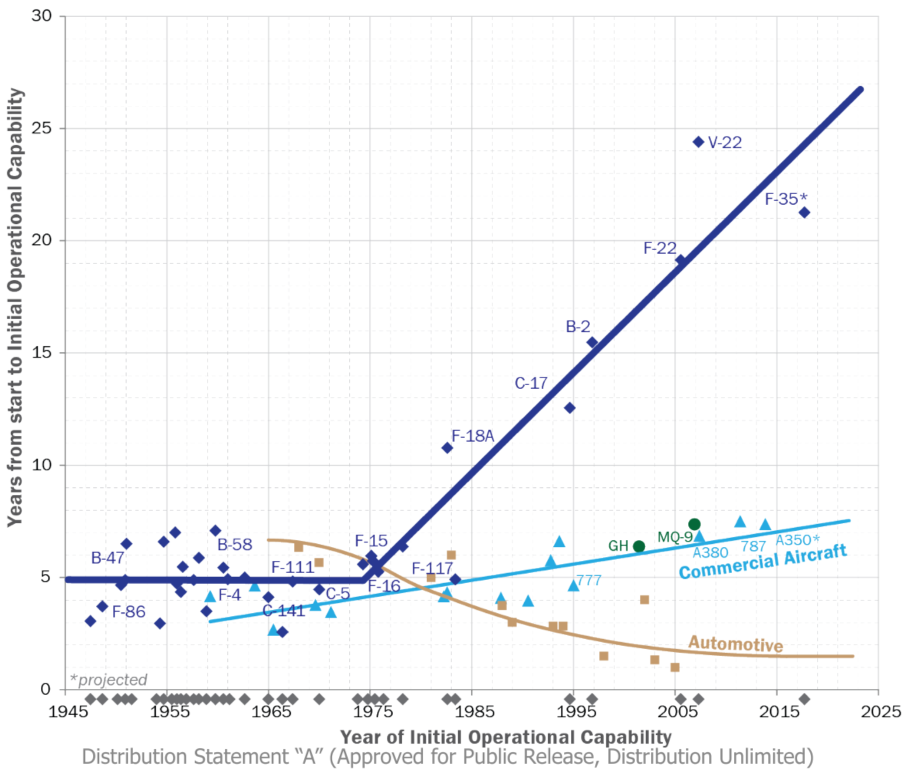


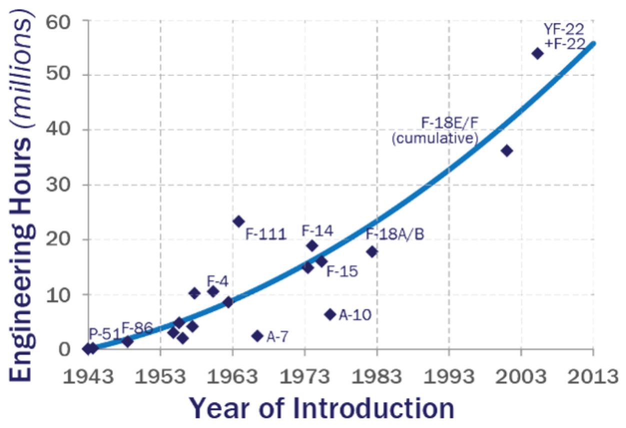
Figure 1-1: Development time of military and commercial aerospace concepts has been increasing over time, in contrast with the reductions in the automotive industry.

transistors we can pack on a chip has increased by roughly four orders of magnitude from 1971 to 1996, but this increased computational power has not correlated with a reduction in engineering hours required in design as expected. And this is only considering hardware improvements, not to mention the developments in optimization and simulation capabilities, and other enabling software. It must either be true that the uptake of new design optimization methods has been low in industry, or that the methods available do not adequately address the underlying challenges in conceptual design. In both cases, Figure 1-2 argues that there is a need to rethink design processes.

1.2 Review of aerospace conceptual design methods

Design stands alone among the many disciplines that aerospace engineering comprises of, e.g., aerodynamics, structures and dynamics, for the reason that it is truly interdisciplinary. Design requires knowledge in all relevant fields, and is about *making decisions with a knowledge of tradeoffs*.

Most aerospace disciplines are technical and technology driven, either discovering fundamental science or making breakthroughs with the aim of expanding the realm of the possible. However, there is a sense that design is as much an art as it is engineering. To



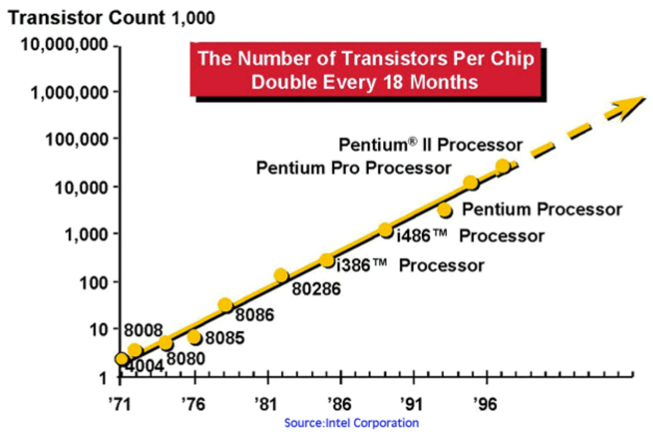


Figure 1-2: Improvements in computation have not correlated with increased efficiency in the development of aerospace industry products.

quote Raymer, a thought leader in design methology, "to some extent good designers ‘are born, not made’" [78]; there aren’t many domains in engineering that would merit a similar assessment. While we don’t believe that good design practices are naturally endowed, it is a fact that a handful of engineers tend to make the majority of key design decisions in aerospace projects. This is because there are often tradeoffs in design that we are (at least at a given time) unable to capture, either because we do not yet understand how to model or optimize them in a computational framework, or when we do, the design problems quickly become intractable. These challenges incentivize leveraging the limited expertise of a small number of design engineers, who use their intuition to make decisions, even when faced with concepts where they have little prior experience.

Increasingly however, new advances in numerical methods and improvements in computation are changing norms in aerospace design. This revolution has been most notable in the increase in accuracy and time efficiency of analysis methods [85], and especially in computational fluid dynamics (CFD). Numerical methods have the potential to improve the design process, by allowing us to better capture complex physics and consider tradeoffs through multidisciplinary design optimization (MDO); they may thus remove the need to rely on imperfect intuitions and democratize the decision making process. Sobieszczanski-Sobieski and Haftka [85] argue that improved computation has resulted in three major thrusts in MDO in aerospace research and industry. We paraphrase these as:

1. **few-disciplinary optimization (FDO),** which focuses on design problems where there are 'two or three interacting disciplines' requiring a narrow scope of expertise to avoid 'organizational challenges' or 'the need for multiobjective optimization'.

2. **conceptual system-level design optimization (CSDO),** which deals with system-level design at the conceptual level using simple, modular analysis tools and/or models, which become more sophisticated during the conceptual design process.

3. **organizational MDO (O-MDO)**, which focuses on organizational coordination challenges by decomposing the system into disciplinary modules and using global sensitivity techniques to couple them, in an attempt to improve data transfer without requiring modifications to existing subsystem design methods. O-MDO is often used to link existing FDO architectures.

1.2.1 A perspective on MDO methods

However, not all MDO approaches are created equal. This section compares CSDO approaches to FDO and O-MDO approaches which are especially prevalent in the aerospace industry, and highlights the relative strengths of CSDO. There is significant literature on different types of optimization problems used in engineering design, which Martins et al. classify broadly as all-at-once (AAO) architectures and distributed architectures [66].

CSDO occurs almost exclusively using AAO architectures which include full coupling of all system variables and constraints. As such, a CSDO model reflects the physics of the real system as closely as practically possible. In contrast, FDO and O-MDO are used in distributed architectures where design problems are partitioned into subproblems and coupled using "complicating constraints" [24]. The primary motivation for decomposing AAO problems "comes from the structure of the engineering-design environment" [66]. The disciplinary groups "may be geographically distributed and may communicate infrequently, (and) typically like to retain control of their own design procedures and make use of in-house expertise" [66].

This is costly in several respects. Firstly, discipline-specific optimizations without ability to perform system-level tradeoffs can result in suboptimal design decisions. A good demonstration is in Figure 1-3. In absence of the ability to concurrently optimize an aircraft and powerplant, aircraft and engine companies can only see the gradient of their objective function (in green) in the degrees of freedom (DOF) that they control. As a result, their domain-specific optimizations produce low-impact locally optimal designs that are far from the true optimum of the system.

There are other practical concerns for implementing FDO approaches on subproblems. In absence of the ability to make system-level decisions, FDO problems contend with incongruity between the DOF available for optimization and the number of constraints, where unsupervised optimization can result in designs that are impractical for real-world applications. Paraphrasing Drela [28], to avoid this pitfall, the number of degrees of freedom

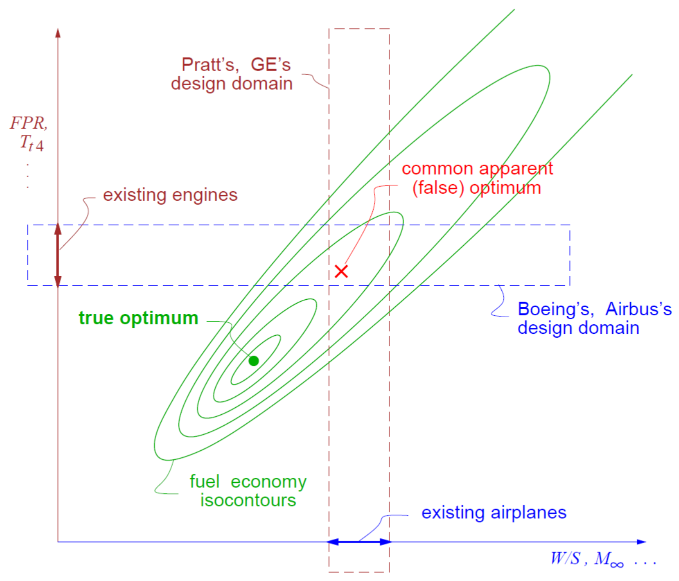


Figure 1-3: Discipline-specific optimization without knowledge of system-level tradeoffs can result in suboptimal design decisions. Figure borrowed from [29].

(i.e. free variables of the system) have to be of similar order as the number of constraints imposed on the system. FDO formulations often have a disproportional number of DOF to the number of constraints or operating conditions because of the inability to observe system-level tradeoffs.

Furthermore, the use of FDO for local optimization without a sufficient understanding of uncertainty can be unproductive. FDO tools, while *accurate* for assessing performance under particular conditions, are subject to a similar level of parametric uncertainty to all MDO tools and are *imprecise* at predicting performance under real conditions. This results in the purported benefit from FDO optimizations often being of the same order of magnitude as the uncertainty in the objective function due to unknown but estimated parameters.

Proponents of O-MDO would argue that O-MDO methodically couples FDO methods to overcome FDO's suboptimality and DOF issues, and that it targets global optimality and design efficiency. O-MDO is the dominant form of optimization in the aerospace industry [5], and the data suggests a weak track record in achieving either of these objectives (see Figures 1-1 and 1-2). Belie, an aerospace industry insider, elaborates on the non-technical barriers for MDO in the aerospace industry in [5] using the cartoon from Figure 1-4, and gives a perspective on how existing O-MDO approaches result in suboptimal systems and organizational outcomes: "Unfortunately the humor reinforces the sad misperception that conflict is a natural feature of product development when, in fact, conflict is more an artifact of the decomposition process itself." [5] The "armies", each with their own hierarchies, create "walls and moats" around their respective disciplines, "isolating communication and fragmenting the solution space as yet another barrier to achieving shared best systems design." [5]

Thus, to achieve real design consensus and global optimality, it is necessary that there is an ability to perform AAO optimization, which can only occur through CSDO architectures

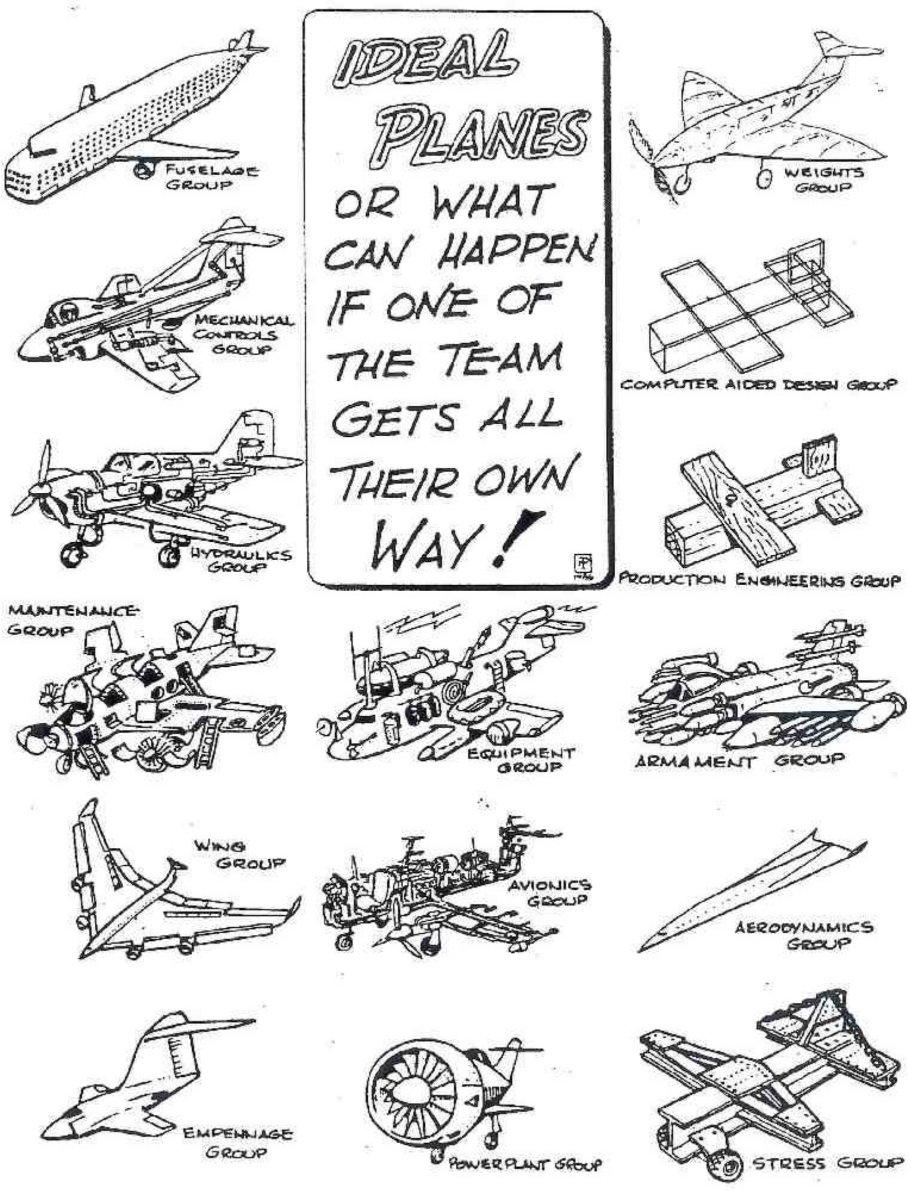


Figure 1-4: O-MDO creates an adversarial design environment through its decomposition approach, creating a push-pull design process.

that are created collaboratively and revised dynamically. CSDO is key for "true integration" as discussed by Agte et al. [1], where the ideal engineered systems are designed in real time, responding to models, analyses and inputs generated by subsystem teams. It is the aim of this thesis to propose methods for engineering design that enable such CSDO at all stages of the design process.

1.3 The mathematical optimization design paradigm

Tractability and practicality of CSDO can be a challenge. Nonlinear optimization methods such as gradient and heuristic methods have made remarkable breakthroughs in MDO, but their scope is limited to local and low-dimensional MDO respectively. An alternate approach is mathematical optimization, i.e. disciplined formulation of engineering design problems using specific mathematical structure. As such, we propose the disciplined formulation of CSDO problems in forms compatible with linear and convex optimization, i.e. efficient mathematical programs[1]. The three mathematical forms of interest are the linear program (LP), the geometric program (GP) and the signomial program (SP), and their mixed integer (MI) counterparts.

The primary benefits of these mathematical programs in engineering design are summarized below:

- **Physics-based design:** The mathematical structure of LPs, GPs and SPs allow clear synergies with design [72]. The language of inequalities allows for designers to have a clear understanding of tradeoffs as they formulate optimization-compatible design models.
- **Mathematical guarantees:** Since LPs and GPs are convex optimization problems, their solutions are guaranteed to be globally optimal. In Chapter 3, their mathematical structures also enable tractable methods for optimization under uncertainty.
- **Sensitivities:** LPs and GPs allow for parameter and constraint sensitivities to be computed at essentially zero computational cost [45].
- **Extensibility:** LPs and GPs can be modular and arbitrarily complex, and many kinds of engineering problems can be cast as LPs or GPs. SPs extend the capabilities of GPs to encompass even more challenging design problems [54].
- **Speed:** The speed of LPs has been demonstrated widely in the solution of large-scale optimization problems. Kirschen [53] quantitatively demonstrates that GPs and SPs are superior to other nonlinear programming methods in both solution quality and speed as well.

In the following sections, we give a formal mathematical background on LPs, GPs and SPs, as well as on mixed integer optimization (MIO).

1.3.1 Mathematical background

We first introduce notation that will be important in the following chapters. Then we refresh the reader on the form of LPs, GPs and SPs; which are optimization methods that we rely on for conceptual design in this thesis.

[1]Program/programming refers to the mathematical formulation of an optimization problem.

Notation

Bold letters (e.g. $\mathbf{x}$) indicate the multi-dimensionality of variables and functions. $[n]$ denotes the set of indices $\{1, \ldots, n\}$. We let $||\cdot||_p$ denote the standard p-norm; if p is not indicated, then it is the Euclidian norm with $p = 2$.

Linear programming

Linear programming is the most fundamental of mathematical programs, and is an optimization problem of the form

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned} \tag{1.1}$$

Since the objective and constraints in a LP are linear, there are some valuable parallels between linear programming and solutions of linear systems. Mainly, the LP solves an underdetermined linear system $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ while minimizing a linear function $\mathbf{c}^\top\mathbf{x}$. Conversely, finding the solution of a full rank linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is equivalent to solving the equivalent LP with a single feasible point.

However, the solution methods for linear systems and LPs are markedly different. Linear systems are generally solved via a series of *elimination* steps, where row operations are performed so that variables are hierarchically eliminated from each row of the $\mathbf{A}$ matrix. The final result is a *row echelon form* of $\mathbf{A}$, which can easily be solved to find the value of $\mathbf{x}$ satisfying $\mathbf{A}\mathbf{x} = \mathbf{b}$.

On the other hand, since a LP is a linear system with an infinite number of solutions, LPs are generally solved via Dantzig's Simplex method, which relies on visiting the extreme points of the polyhedron defined by the constraint $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ in a sequence of *pivot operations*, i.e. moves between basic solutions across an edge of the feasible polyhedron. The algorithm terminates when there are no adjacent basic solutions to the incumbent with lower cost [14]. In the worst case, the Simplex method takes exponential-time, since the number of extreme points of a polyhedron increases exponentially with the number of variables and constraints. However, the worst case performance is only seen in rare degenerate cases, and in practice the method can solve large-scale linear programs with large numbers (10^6) of variables and constraints in minutes on unremarkable personal computers.

Geometric programming

Geometric programming is a method of log-convex optimization that has been developed to solve problems in engineering design [31]. A *geometric program in posynomial form* is a log-convex optimization problem,

$$\begin{aligned} \min \quad & f_0(\mathbf{u}) \\ \text{s.t.} \quad & f_i(\mathbf{u}) \leq 1, \; i \in [m_p], \\ & h_i(\mathbf{u}) = 1, \; i \in [m_e], \\ & \mathbf{u} \in \mathbb{R}^n_{++}, \end{aligned} \tag{1.2}$$

where each f_i is a *posynomial*, each h_i is a *monomial*, m_p is the number of posynomials, and m_e is the number of monomials. A monomial $h_i(\mathbf{u})$ is a function of the form

$$h_i(\mathbf{u}) = e^{b_i} \textstyle\prod_{j=1}^{n} {u_j}^{a_{ij}}, \tag{1.3}$$

where a_{ij} is the j^{th} component of a row vector $\mathbf{a_i}$ in $\mathbb{R}^n$, u_j is the j^{th} component of a column vector $\mathbf{u}$ in $\mathbb{R}^n_{++}$, and b_i is in $\mathbb{R}$. An example of a monomial is the lift equation, $L = \frac{1}{2}\rho V^2 C_L S$. A posynomial $f_i(\mathbf{u})$ is the sum of $K \in \mathbb{Z}^+$ monomials,

$$f_i(\mathbf{u}) = \textstyle\sum_{k=1}^{K} e^{b_{ik}} \prod_{j=1}^{n} {u_j}^{a_{ikj}}, \tag{1.4}$$

where a_{ikj} is the j^{th} component of a row vector $\mathbf{a_{ik}}$ in $\mathbb{R}^n$, u_j is the j^{th} component of a column vector $\mathbf{u}$ in $\mathbb{R}^n_{++}$, and b_{ik} is in $\mathbb{R}$ [16]. The stagnation pressure definition is a good example of a posynomial: $P_t = P + \frac{1}{2}\rho V^2$.

A logarithmic change of the variables $x_j = \log(u_j)$ would turn a monomial into *the exponential of an affine function* and a posynomial into *the sum of exponentials of affine functions.* A transformed monomial $h_i(\mathbf{x})$ is of the form

$$h_i(\mathbf{x}) = e^{\mathbf{a_i}^\top \mathbf{x} + b_i}, \tag{1.5}$$

where $\mathbf{x}$ is a column vector in $\mathbb{R}^n$. A transformed posynomial $f_i(\mathbf{x})$ is the sum of $K_i \in \mathbb{Z}^+$ monomials,

$$f_i(\mathbf{x}) = \textstyle\sum_{k=1}^{K_i} e^{\mathbf{a_{ik}}^\top \mathbf{x} + b_{ik}}, \tag{1.6}$$

where $\mathbf{x}$ is a column vector in $\mathbb{R}^n$. A geometric program with transformed constraints is a *geometric program in exponential form*, and is a convex optimization problem. Geometric programming in exponential form also has an interesting parallel with linear programming, mainly that the logarithm of monomials is linear: $\log(h_i(\mathbf{x})) = \mathbf{a_i}^\top \mathbf{x} + b_i$. This is why monomials can be used in equality constraints as well as inequalities.

Although theory of the GP has existed since the 1960's, GPs have recently experienced a resurgence due to the advent of polynomial-time interior point methods [70] and improvements in computing. Interior point methods transform the constrained convex optimization problem into an unconstrained convex optimization problem by introducing *barrier functions*, which add penalties to constraint violation in the objective function. Then, starting from an initial feasible point, they apply Newton's method to the unconstrained optimization problem until the Karush-Kuhn-Tucker (KKT) conditions are met, guaranteeing global optimality.

GPs are particularly attractive for engineering design because the constraints comply with power law forms. Boyd's tutorial on GPs [16] contains a number of examples of engineering problems that are compatible with geometric programming. Since many engineering problems of interest have strictly positive variables, GPs are quite applicable, and certain variable transformations can make problems with negative quantities tractable. GPs have been effective in addressing a subset of aircraft conceptual design problems [45, 20, 73]. However, the restriction of posynomials to the *less-than-side of inequalities* is a significant barrier to the use of GPs in other contexts, and motivates the introduction of signomials.

Signomial programming

The SP is the difference-of-log-convex extension of the GP, albeit with the loss of some mathematical guarantees compared to the GP [54]. A *signomial* can be defined as the difference of two posynomials. Consequently, a SP is a non-log-convex optimization problem,

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{u}) \\ \text{s.t.} \quad & f_i(\mathbf{u}) - g_i(\mathbf{u}) \leq 0,\ i \in [m], \\ & \mathbf{u} \in \mathbb{R}^n_{++}, \end{aligned} \tag{1.7}$$

where f_0, f_i and g_i are all posynomials. Reliably solving a SP to a local optimum has been described in [16] and [59]. A common solution heuristic involves solving a SP as a sequence of GPs, where each GP is a local log-convex approximation of the SP. Signomial programming has been used to great effect in modeling and designing complex aircraft at a conceptual level quickly and reliably [94, 54, 52].

Mixed-integer optimization

The above optimization formulations can be extended to include *integer variables* in $\mathbf{x}$,

$$\mathbf{x} \in \mathbf{X} \subseteq \{\mathbb{Z}^k \cup \mathbb{R}^{n-k}\}. \tag{1.8}$$

An optimization problem is called a MIO problem when it has a combination of continuous and integer variables. The most common example of a MIO problem is the mixed integer linear optimization (MILO),

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbf{X} \subseteq \{\mathbb{Z}^k \cup \mathbb{R}^{n-k}\}. \end{aligned} \tag{1.9}$$

In MIO, many integer variables commonly show up as *binary variables*, which are restricted to the $\{0, 1\}$ domain. Binary variables are most often used to indicate logical constraints, such as but not limited to *and*, *or*, or *if-then* relationships. Integer variables otherwise show up in design problems where we must select from among a set of discrete options, such as materials, components or configurations.

A MILO is more difficult to solve than its linear relaxation, because the extreme points of the polyhedron described by the constraint $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ are non-integral in the integer variables with high probability. The other cases are so-called *locally ideal formulations*; Vielma defines a MILO formulation as locally ideal if and only if its linear relaxation has a basic feasible solution and all basic feasible solutions are integral in the integer variables [91].

In general, MIO problems are solved using *branch-and-bound* methods. These methods first solve a continuous relaxation of the original optimization problem. Then, the solution is repaired to integrality through a series of hierarchical *branching operations*, building a *branch-and-bound tree.* New constraints are added to the relaxation at each branch of the tree to restore integrality to subsets of integer variables. The optimal solution of each relaxation at a tree node is a *lower bound* on the solutions of branches originating from that node. New integral basic feasible solutions throughout the tree are *upper bounds* on the true optimum, which are used to prune the branches of the branch-and-bound tree to reach faster convergence. The optimum is reached when we have pruned or processed all branches and found the objective-minimizing leaf node.

1.4 Improving conceptual design through optimization

Mathematical optimization has been implemented for CSDO in many engineering fields with great success. There is a wealth of literature on the applications of convex and mixed-integer convex optimization to fields as diverse as signal processing [60], microgrid energy management [40], and guidance and control of aerospace vehicles [64]. Boyd et al. summarize a variety of successful applications of geometric programming and its extensions in engineering [16].

However, there are new developments in mathematical programming that have flourished in the operations research and optimization literature, but have yet to update and upgrade existing conceptual engineering design methods. Thus, many cutting edge methods in optimization have yet to be translated to improvements in conceptual design process. This thesis is an interdisciplinary work in engineering and optimization that bridges the gap. We propose leveraging several developments in the fields of *mixed-integer optimization*, *machine learning*, and *robust optimization* to address two opportunities to extend the capabilities of MDO methods.

The first challenge in MDO we address is the ability to more effectively incorporate arbitrary constraints, black boxes and data into optimization models. One of the disadvantages of the mathematical optimization design paradigm is the requirement for constraints to be of compatible mathematical forms, and more specifically to be restricted to a specific set of mathematical primitives. This precludes the use of constraints that are inefficient (e.g. nonconvex), inexplicit (e.g. results of simulations), or data-driven (e.g. results of past simulations or experiments). Conceptual design is the phase of the design process that has the greatest impact on the success of an engineering project, and it is also the phase where we must reconcile the greatest variety of models. It is thus critical to improve the compatibility of efficient optimization with general explicit and inexplicit constraints.

A majority of approaches in the literature address this challenge through *parameter learning*, i.e. function fitting. More specifically, they fit the underlying constraints or data with more mathematically efficient representations, albeit with some loss of accuracy. Some good examples of parameter learning are work by Magnani et al. [63] and Hoburg et al. [44], who have addressed the problems of convex polynomial and posynomial fitting respectively, as a method to incorporate functions or data into efficient optimization. However, these methods assume that the underlying function has a convexity or log-convexity property, which is a strong assumption given that many real-world functions do not exhibit such behavior. Some examples of explicit nonconvex functions that are commonly found in nature are sigmoids, sinusoids and the logarithm. In addition, the methods consider parameter learning without the potential for efficient design of experiments in the learning process.

To improve our ability to include arbitrary constraints in a MDO, or more generally a global optimization framework, we propose a more general *constraint learning* framework that leverages modern machine learning (ML) techniques. Using optimal decision trees, and a suite of existing and novel sampling techniques, we generate optimization-compatible constraints from arbitrary constraints, black boxes or data. These surrogates are MIO-compatible, and thus allow for the incorporation of aforementioned classes of intractable constraints into efficient optimization, without making assumptions on the forms of the underlying constraints. We are then able to solve these MIO approximations, and repair them to find optimal solutions to the original design problems.

The second challenge is the ability to tractably and deterministically consider uncertainty in the design process. Few design optimization tools in the aerospace literature can consider risk due to parametric uncertainty. A 2018 review by Papageorgiou et al. [75] lists a few such tools under the umbrella of *non-deterministic* approaches, i.e. *stochastic optimization*. These state-of-the-art stochastic methods assume that uncertain parameters come from known probability density functions, and propagate these functions through physical models of the engineered system to determine their effects on constraint feasibility and the objective function. However, these methods have been shown to be impractical and intractable for addressing real-world problems, and make the unrealistic assumption of known probability distributions. Perhaps more importantly however, these approaches are non-deterministic by definition; it is unsatisfactory for aerospace engineers to arrive at different optimal designs depending on the random outcomes of uncertain parameters from unknown but assumed distributions.

In this thesis, we propose addressing engineering design problems under uncertainty via *robust optimization (RO)*, i.e. optimization over parameters belonging in an uncertainty set. RO has several important advantages over stochastic optimization. Instead of requiring that uncertain parameters come from known probability distributions, RO requires the milder assumption that the parameters belong in an uncertainty set, e.g., ellipsoidal or box sets. Using the robust counterpart of the design problem, RO can address design under uncertainty in a deterministic manner. Specifically, the optimal design is one that satisfies all constraints under all outcomes of parameters from the uncertainty set, while minimizing the worst-case objective.

The use of RO in other engineering fields is not a new development by any means; notable applications from structural optimization to circuit design are summarized in [8]. However, aerospace system design under uncertainty requires the formulation of robust counterparts for optimization problems that adequately capture aerospace physics. To design aerospace systems deterministically under uncertainty, we propose the formulation of robust signomial programs, which have shown promise in the design optimization of aerospace systems [94, 54, 52]. The proposed robust signomial programming framework can address a variety of engineering design problems, and will increase confidence in the ability of systems to satisfy constraints under uncertain parameter outcomes. By extension, such a rigorous integration of uncertain parameters into design methods will improve trust in design tools.

With the methods from this thesis, engineers will have the potential to consider system-level tradeoffs under a general, tractable and practical framework, leveraging modern mixed-integer optimization, machine learning and robust optimization techniques that have matured outside of the science and engineering literature.

1.5 Thesis objectives and outline

In summary, we propose to investigate the following two key questions for CSDO methods:

- How do we generate tractable optimization models that can consider arbitrary explicit and inexplicit constraints that describe the physical world?
- How do we tractably and deterministically optimize systems that are robust against uncertain outcomes?

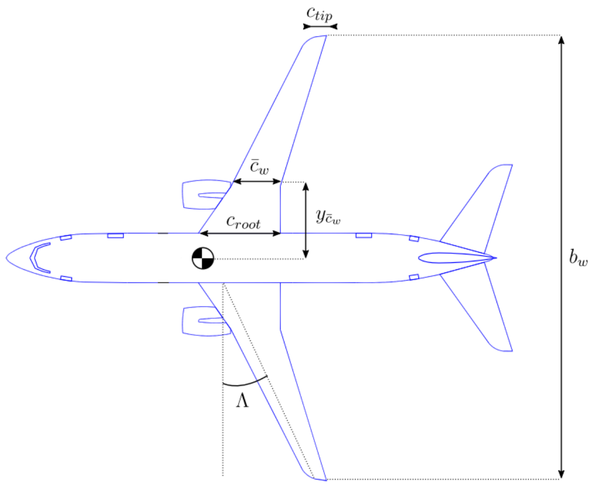
Though we focus primarily on applications in aerospace engineering, the proposed methods are general to a wide array of design and decision making problems. Chapter 2 introduces constraint learning via optimal decision trees, enabling global optimization over general explicit and inexplicit constraints. The only requirement for the proposed method is a bounded domain over the decision variables in learned constraints. Chapter 3 describes the formulation of robust signomial programs, a scalable and deterministic RO approach to designing aerospace systems under parametric uncertainty.

Chapters 2 and 3 are predominantly standalone. They motivate and demonstrate the need for the proposed methods, and review the state-of-the-art in academic literature. They provide the requisite mathematical knowledge before proposing new methods to tackle each design optimization challenge. The chapters describe the methods in complete detail before applying them to a number of benchmark and real-world problems. The software implementations of the two methods can be found via links in Appendices A.1 and B.1. This thesis develops interdisciplinary work that meshes developments in optimization, operations research and machine learning with engineering design, with contributions in all four fields.

Chapter 2

Global Optimization via Optimal Decision Trees

Engineering design is often informed by a combination of *explicit* and *inexplicit* constraints. We define explicit constraints as those that can be represented in closed form, i.e. using a finite number of directly computable mathematical primitives. These are the ones most commonly faced in the early phases of conceptual design. An example explicit constraint acts on the aircraft in Figure 2-1, and is the maximum wing root bending moment constraint from [94]. The constraint states that the maximum moment at the wing root is greater than or equal to the maximum moments due to lift and weight forces integrated over the wing. This ensures that the wing root can support the most strenuous forces during steady flight.



$$M_r c_{\text{root}} \geq \Big(L_{\text{wing,max}} - N_{\text{lift}}(W_{\text{wing}} + f_{\text{fuel,wing}} W_{\text{fuel,total}})\Big)\left(\frac{b_w^2}{12 S_w}(c_{\text{root}} + c_{\text{tip}})\right) - N_{\text{lift}} W_{\text{engine}} y_{\text{engine}}$$

Figure 2-1: Maximum wing root bending moment of a commercial aircraft, from [94]. An explicit constraint.

Inexplicit constraints do not have closed form representations. A good aerospace example is drag polars, as shown in Figure 2-2, which describe the relationship between the coefficients of lift and drag of an airfoil at different Reynolds numbers. These physical relationships are described by the Navier-Stokes equations for the motion of viscous fluids. Since these relationships are simulated approximately via XFOIL [27], a black box tool, they have no analytical representations.

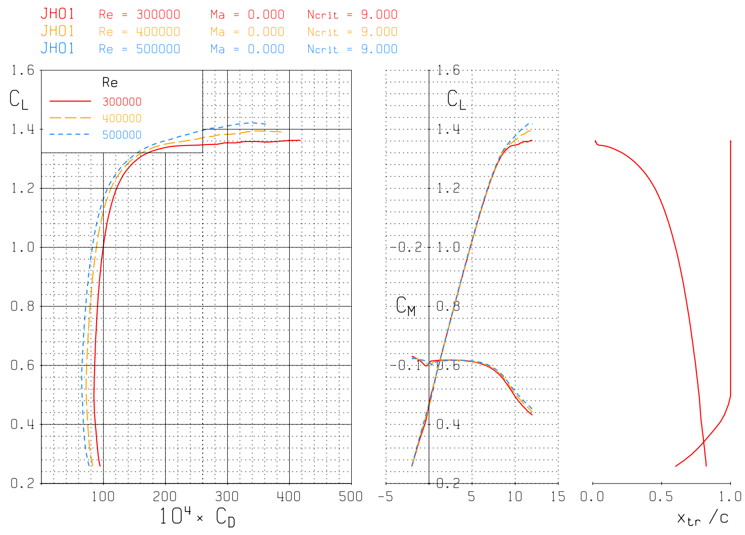


Figure 2-2: Drag polars showing the relationship between lift, drag and moment coefficients, and Reynolds number, as well as the transition location for an airfoil. An inexplicit constraint. Figure borrowed from [73].

Both explicit and inexplicit constraints can pose significant challenges for optimization of real-world systems, since they may not conform to efficient mathematical forms. Explicit constraints may be nonlinear and nonconvex, meaning that they cannot be addressed by linear and convex optimization methods which have guarantees of global optimality. While many inexplicit constraints exhibit convex behavior and are well approximated by convex functions [62, 44], this is in general a strong assumption, and many inexplicit systems have outputs of interest that are nonlinear and nonconvex functions of the input data.

Optimization over general constraints and objective functions is called *global optimization*. The meaning of *global* is two-fold. Most explicitly, global optimization problems look for *global optima* over the feasible set of decision variables. But additionally, the optimizers are global in the sense of *generality*, i.e. having the ability to be applied to constraints and objective functions with arbitrary mathematical primitives. This is where most existing global optimizers falter, by placing restrictions on the types of constraints allowed. This chapter aims to improve the state of the art in global optimization by using modern ML methods for constraint learning, especially to derive optimization-compatible (i.e., mixed integer linear or convex) constraints from arbitrary constraints, models or data.

2.1 Review of global optimization

More formally, global optimization seeks to address the following problem,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \geq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}, \\ & \mathbf{x} \in \mathbb{Z}^m \times \mathbb{R}^{n-m}, \end{aligned} \tag{2.1}$$

where f, $\mathbf{g}$ and $\mathbf{h}$ are the objective function, inequality constraints and equality constraints, respectively, and $\mathbf{x}$ is a vector of decision variables. The objective functions and constraints may or may not conform to any specific mathematical structure, unlike linear or convex optimization problems, and variables can be continuous or integer.

Existing global optimizers approximate problem (2.1) into forms compatible with efficient optimization. These optimizers use three major approaches, which are gradient-based methods, outer approximations, and MIO methods. The gradient-based approach is used by popular nonlinear solvers such as CONOPT and IPOPT. These solvers initialize their solution procedure using feasible solutions found via efficient heuristics. Then, they solve a series of gradient descent iterations, confirming optimality via satisfaction of the KKT conditions. As detailed by Drud [30], CONOPT relies on a generalized reduced-gradient algorithm, linearizing the constraints and solving a sequence of linear-searching gradient steps, maintaning feasibility to tolerance at each step. Wachter and Biegler [92] describe IPOPT's primal-dual barrier approach. It relaxes the constrained global optimization problem into an unconstrained optimization problem using a logarithmic barrier function, then uses a damped Newton's method to reduce the optimality gap to a desired tolerance. These gradient-based optimizers are efficient and effective in the presence of nonlinear constraints that are sparse, being able to solve problems on the order of 1000 variables and constraints on unremarkable personal computers in minutes to local optimality.

Another approach is an outer approximation approach, described by Horst et al. [46]. This approach simplifies a global optimization problem by approximating constraints via linear and nonlinear cuts that preserve the original the feasible set over decision variables $\mathbf{x}$. This approach is effective for constraints with certain mathematical structure (e.g., linearity of integer variables and convexity of nonlinear functions considered by Duran and Grossmann [34], or concavity or bilinearity of constraints considered by Bergamini et al. [7]), where mathematically efficient outer approximators exist. While these approaches are effective, they have found less commercial success due to their problem specific nature.

A final approach, and one that meshes naturally with optimization over integer variables, couples MIO with outer approximations. Ryoo and Sahinidis [79] present an impactful approach called the branch-and-reduce method, which relies on recursively partitioning the domain of each constraint and objective over the decision variables, and bounding their values in each subdomain by examining their mathematical primitives. Such recursive partitioning creates a branch-and-bound tree, the solution to which has guarantees of global optimality through the bounding and pruning process inherent in solving MIO problems via branch-and-bound. This method has seen success in BARON [81], a popular commercial global optimizer.

While the aforementioned approaches are effective in addressing certain classes of global optimization problems, each of these approaches has weaknesses. In general, gradient-based approaches rely on good initial feasible solutions, and are ineffective in presence of integer decision variables. Outer approximation approaches fail to generalize to global optimization problems with general nonlinearities. While being more general than outer approximation methods, existing MIO approaches don't scale as well due to their combinatorial nature.

Perhaps more importantly, in pursuit of mathematical efficiency, many global optimizers place additional constraints on the forms of constraints, requiring constraints to use a small subset of possible mathematical primitives. For example, BARON "can handle functions that involve $\exp(x)$, $\ln(x)$, x^{α} for real α, and β^{x} for real β" [81]. Constraints from the real world do not always adhere to these forms, and often involve other classes of functions such as trigonometric functions, signomials, and piecewise-discontinuous functions. It is often not possible to transform these functions into forms compatible with existing global optimizers. These optimizers face even greater challenges when dealing with objectives and constraints that are black box. Black box constraints are *inexplicit*, meaning that they have no analytical representations, such as when constraints are the outcomes of simulations.

In this chapter, we propose a new approach to reformulate global optimization problems as MIO problems using ML, leveraging work by Bertsimas and Dunn [10, 11] on the optimal classification tree with hyperplanes (OCT-H) and the optimal regression tree with hyperplanes (ORT-H). The approach addresses global optimization with arbitrary explicit and inexplicit constraints. The only requirement for the proposed method is a bounded feasible domain for the subset of decision variables $\mathbf{x}$ present in nonlinear constraints.

In our proposed method, we approximate each constraint that is outside of the scope of efficient mathematical optimization using an OCT-H. More specifically, each nonlinear constraint $g_i(\mathbf{x}) \geq 0$ is approximated by an OCT-H T_i trained on data $\{(\tilde{\mathbf{x}}_k, \mathbb{I}(g_i(\tilde{\mathbf{x}}_k) \geq 0)),\ k \in [n]\}$, where $\tilde{\mathbf{x}}_k$ is an outcome of decision variables, $\mathbb{I}$ is the indicator function, and $g_i(\tilde{\mathbf{x}}_k)$ is the left-hand-side of the constraint evaluated at $\tilde{\mathbf{x}}_k$. Thus, tree T_i makes an approximation of the feasible space of constraint $g_i(\mathbf{x}) \geq 0$, predicting (with some error) whether an outcome of decision variables satisfies the constraint. This approach also extends to approximate each nonlinear equality $h_j(\mathbf{x}) = 0$, and approximates nonlinear objective functions via ORT-Hs.

The approximating trees allow for a natural MIO approximation of the underlying constraints. Each feasible leaf of an OCT-H is reached by a decision path defining an intersection of halfspaces, i.e. a polyhedron. Constraints may thus be approximated as a union of feasible polyhedra of the approximating OCT-Hs using disjunctive constraints. We solve this efficient MIO approximation of the original problem to obtain a near-feasible and near-optimal solution, and then use gradient-based methods to repair the solution to be feasible and locally optimal.

The proposed method has several strengths relative to other global optimization methods. It is agnostic of the forms of constraints in the problem; as long as we can query whether a sample $\tilde{\mathbf{x}}$ is feasible to a constraint, we can embed the constraint into the MIO approximation. Once the constraints are learned using decision trees, the solution time of the resulting MIO approximation is low compared to solving the original global optimization problem. The proposed method can also be used to generate constraints from data which may not come from any known function, simulation or distribution. This allows us to simultaneously learn the physics of complex phenomena such as but not limited to social dynamical models or solutions of partial differential equations, and embed them into optimization problems.

In this chapter, we present our global optimization approach, implemented in our optimizer OCT-H for Global Optimization (OCT-HaGOn), pronounced "octagon". We demonstrate its promise by considering global optimization problems with explicit nonlinear constraints. This allows us to quantify the performance of our method against existing global optimizers using available benchmarks. In addition, we approximate all nonlinear constraints in the benchmarks regardless of their efficient optimization-representability. The proposed method extends to mixed-integer-convex approaches where we embed efficiently-optimizable nonlinear constraints (e.g., quadratic, second order conic, log-sum-exponential constraints) into the MIO formulation directly, as long as these constraints are supported by the underlying solver.

2.1.1 Role of machine learning in optimization

The role of optimization in training ML models is well known and studied. Recent review papers in the literature [38, 86] survey the landscape of mathematical optimization and heuristic methods used for a variety of ML applications. However, we are interested in the inverse of the above, and specifically how ML can be used for the purpose of optimization, especially to solve problems that cannot naturally be posed as efficient optimization problems.

There is precedent for using ML methods to improve computational efficiency. A prominent example is the use of ML to accelerate the simulation of nonlinear systems such as those in computational fluid dynamics [55], molecular dynamics [39] and quantum mechanics [69]. There has been some prior work using ML to accelerate optimizations, e.g., using Bayesian optimization [35] or neural networks [87]. While these show that ML-driven optimization is theoretically possible, the proposed methods are computationally expensive and not scalable for real-world problems. An interesting parallel use of ML in optimization is in the interpretation of optimal solutions, where ML is used to understand the optimal strategies (i.e. outcomes of all or subsets of decision variables) resulting from an optimization problem under different parameters [9].

In this work, we use ML to find optimal solutions to global optimization problems involving both explicit constraints with arbitrary mathematical primitives, and inexplicit black box functions. For this purpose, ML is used for *constraint learning* within two capacities. The first capacity is to accelerate optimizations over known models. When models and/or constraints are known but their use is prohibitive, e.g. in the case of explicit but nonlinear and nonconvex constraints, learners are used to create surrogates that are more efficient for use in optimization. The second is in modeling. When data is available but models and/or constraints are black box, learners act as interpolants to the data, and to allow patterns in the data to be embedded in optimization.

Using ML in optimization in both of these capacities requires that the approximating ML models are optimization-representable. While many types of ML models are efficiently queried and accurate, e.g., many types of neural networks and Gaussian processes, they cannot be embedded explicitly into structured optimization. Prior work has recognized the potential for using constraint learning approaches in optimization over data-driven constraints. Both Biggs et al. [15] and Mišić [68] use the prediction of tree ensembles as the objective function of optimization problems, given that a subset of tree features are decision variables. Maragno et al. [65] go further and present a more general approach for data-driven optimization that leverages decision trees as well as other MIO-compatible ML models such as support vector machines and neural networks.

The aforementioned applications of decision trees in optimization are restricted in scope. [15] and [68] limit their applications to optimization over data-driven objective functions, where decision trees are used to regress on a continuous quantity of interest. And while Maragno et al. [65] use constraint learning for data-driven constraints, we use constraint learning to make approximations of intractable explicit and inexplicit constraints as well, where we have the capacity to sample the underlying constraints to generate data. Thus we propose a global optimization framework that can accommodate arbitrary explicit, inexplicit and data-driven constraints, leveraging decision trees in regression and classification settings.

While it is possible to use other MIO-compatible ML models for constraint learning in global optimization as proposed by Maragno et al. [65], we choose to rely on ORT-Hs and OCT-Hs since they are tunable, accurate and interpretable [11]. In the following sections, we demonstrate that a global optimization method leveraging optimal decision trees makes significant progress in using ML for both acceleration of optimizations and modeling, using the natural and intuitive MIO representation of trees.

2.2 Review of decision trees

Decision trees is a popular predictive ML method that partitions data hierarchically according to its features. A class label in a finite set of possible labels is assigned to each leaf node of the tree depending on the most common label of the data falling into the node. This capability is demonstrated in Figure 2-3, where two different decision trees are trained on a data set with $n = 150$ points and $m = 4$ features. The top tree with *axis-aligned splits* is called an *optimal classification tree (OCT)*, and the bottom tree with *hyperplane splits* is called an *OCT-H*. The three colors indicate the actual class $y_i,\ i = [1, 2, 3]$ that each data point corresponds to, where the axes x_1 and x_2 show the values of the first two features. The partitions in the middle plots indicate how the points are classified according to the trees. Note that it is possible and probable that there is some misclassification error on the training data. Notably, all points in green have been properly classified by both models, however, the blue and orange points have some margin of error.

The optimization problem that is solved to produce a decision tree $T \in \mathbb{T}$ over known data $(\mathbf{x}, \mathbf{y})$ is the following:

$$\min_{T} \text{error}(T, \mathbf{x}, \mathbf{y}) + c_p \cdot \text{complexity}(T),$$

where c_p is a complexity penalty parameter which attempts to strike a balance between the misclassification error over the test data and complexity (depth and breadth) of the tree. Once trained, decision trees are queried to predict the classes of test points with known features, but unknown class. In this particular case, a test point $\tilde{x} = [2, 1]$ (red asterisk) would be classified as green in the top tree, whereas in the bottom tree it would be classified

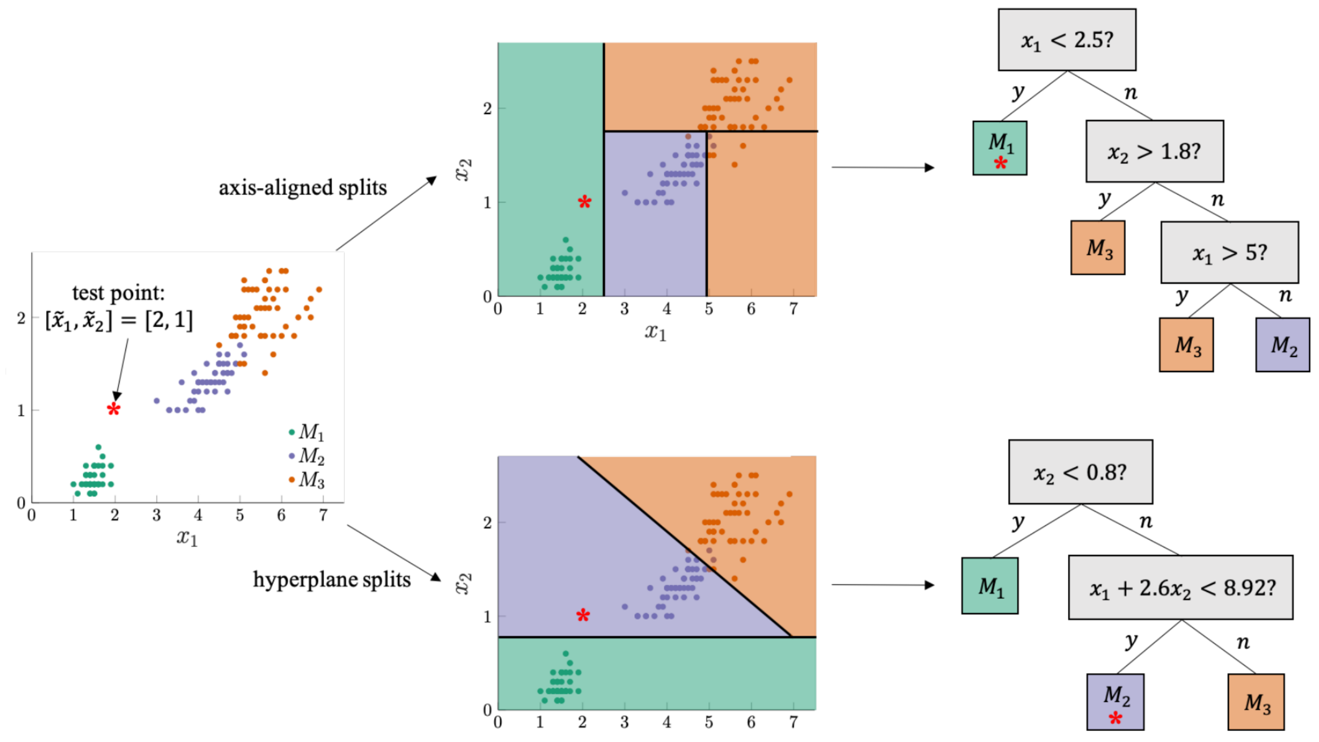


Figure 2-3: Two decision trees classifying the Iris dataset using axis-aligned and hyperplane splits. Borrowed from [50].

as blue.

Decision trees were pioneered by Breiman et al. with the advent of classification and regression trees (CART) [17]. However, CART is a top-down, greedy method of producing decision trees. Each split is only locally optimal since the splits are made recursively on the children of each new split starting from the root node. The ability of decision trees to explore the feature space has improved with the work of Bertsimas and Dunn [32, 11] on the OCT. OCTs leverage MIO and local search heuristics to reduce misclassification error relative to CART without overfitting. Furthermore, OCTs are more *interpretable*, since they can achieve similar mean squared error (MSE) error as trees generated by CART with much less complexity.

OCT-Hs generalize OCTs by allowing for hyperplane splits, i.e. splits in more than one feature at a time. An OCT-H can solve classification problems with higher accuracy and lower complexity than an OCT [11], and is more expressive in an optimization setting due to couplings of decision variables in nonlinear constraints. Thus, our method leverages OCT-Hs exclusively to approximate constraints.

ORT-Hs extend OCT-Hs to regression problems, where the prediction of interest is continuous, i.e. $\tilde{y} \in \mathbb{R}$. Each leaf of an ORT-H, instead of containing a fixed class prediction, contains a continuous prediction $\tilde{y}$ as a linear regression over $\mathbf{x}$ in the domain of the leaf. ORT-Hs are particularly useful when approximating nonlinear objective functions.

We rely on software from the company Interpretable AI (IAI) in building, training and storing problem data in the form of OCT-Hs and ORT-Hs [48].

2.3 Contributions

In this thesis, we propose a global optimization approach that generalizes to explicit and inexplicit constraints and objective functions over bounded dom($\mathbf{x}$). Our specific contributions are as follows:

1. We introduce an ensemble of methods for sampling constraints efficiently for the purpose of constraint learning. We leverage synergies of existing design of experiments (DoE) techniques, but also devise a new k-Nearest Neighbors (kNN) based sampling technique for sampling near-feasible points of explicit and inexplicit constraints.

2. We learn the feasible space of nonlinear objectives, inequalities and equalities using OCT-Hs and ORT-Hs.

3. We make MIO approximations of global optimization problems using the disjunctive representations of decision trees, and solve them using MIO solvers.

4. We devise a projected gradient descent method to check and repair the near-feasible, and near-optimal solutions from the MIO approximations.

5. We apply our method to a set of benchmark and real-world problems, and demonstrate its performance in finding global optima.

In Section 2.4, we detail our method, followed by a demonstrative example in Section 2.5. In Section 2.6, we test our method on a number of benchmark problems from the literature, and compare our results with state-of-the-art global optimization tools such as BARON, IPOPT and CONOPT. In Section 2.7, we use our method to optimize two aerospace systems, one of which cannot be addressed via existing global optimization tools. In Section 2.8, we discuss the results, the limitations of the method, and avenues for future research. Section 2.9 concludes by summarizing our findings and contributions.

2.4 Method

As aforementioned, our goal is to solve the global optimization problem approximately by making an OCT-H based MIO approximation, and then repairing the solution to be feasible and locally optimal. As an overview of this section, our method takes the following steps:

1. **Generate standard form problem:** In order to reduce the global optimization problem to a tractable MIO problem, we first restructure the global optimization problem in (2.1). The linear constraints are passed directly to the MIO problem, while the nonlinear constraints are approximated in steps 2-6 below. If any variables involved in nonlinear constraints are unbounded from above and/or below, we attempt to compute bounds for the purpose of sampling.

2. **Sample and evaluate nonlinear constraints:** The data used in training is important for the accuracy of ML models. For accurate OCT-H approximations of nonlinear constraints, we use fast heuristics and DoE methods to sample variables over $\mathrm{dom}(\mathbf{x})$. We evaluate each constraint over the samples, and resample to find additional points near the constraint boundary for local approximation refinement.

3. **Train decision trees over constraint data**: The feasibility space of each constraint is classified and approximated by an OCT-H. If the objective function is nonlinear, it is regressed and approximated via an ORT-H.

4. **Generate MI approximation:** The decision paths and hyperplane splits are extracted from the trees, and used to formulate efficient MIO approximations of the nonlinear constraints using disjunctions.

5. **Solve MIO approximation:** The resulting MIO problem is optimized using commercial solvers to get an approximate solution.

6. **Check and repair solution:** The MIO problem approximates the global optimization problem, so the optimum is likely to be near-optimal and near-feasible. We evaluate the feasibility of each nonlinear constraint, and compute the gradients of the objective and nonlinear constraints using automatic differentiation. In case of suboptimality or infeasibility, we perform a number of projected gradient descent steps to repair the solution, so that it is feasible and locally optimal.

We describe the steps in greater detail in Sections 2.4.1 through 2.4.6. A step-by-step demonstration of the method, as implemented in our optimizer OCT-HaGOn, can be found in Section 2.5.

2.4.1 Standard form problem

We restructure the global optimization problem posed in (2.1) by separating the linear and nonlinear constraints. The linear constraints are passed directly into a MIO model, while the nonlinear constraints are stored for approximation. If constraints are black box, they are assumed to be nonlinear as well. This restructured problem is shown in (2.2), and referred to as the standard form. Note that the standard form allows for both nonlinear inequalities and equalities.

$$\begin{aligned} \min_{x} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \geq 0, \ i \in I, \\ & h_j(\mathbf{x}) = 0, \ j \in J, \\ & \mathbf{Ax} \geq \mathbf{b}, \ \mathbf{Cx} = \mathbf{d}, \\ & x_k \in [\underline{x}_k, \overline{x}_k], \ k \in [n]. \end{aligned} \tag{2.2}$$

Variable outer-bounding

The proposed method requires boundedness of decision variables $\mathbf{x}$ in each approximated constraint so that we can sample dom($\mathbf{x}$) for constraint evaluation. When bounds are missing for any variable x_k in a nonlinear constraint, we pose the following optimization problem over the linear constraints only.

$$\begin{aligned} \min_{x}/\max \quad & x_k \\ \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{b}, \ \mathbf{Cx} = \mathbf{d} \\ & x_i \in [\underline{x}_i, \overline{x}_i], \ i \subseteq [n]. \end{aligned} \tag{2.3}$$

The solution to this problem is the absolute largest range $[\underline{x}_k, \overline{x}_k]$ that satisfies all linear constraints as well as bounds on x_i, for those indices i for which x_i is bounded. We can also solve the above optimization problem to tighten bounds on variables with existing bounds. Tighter bounds can significantly improve solution quality and time by improving the quality of ML approximations.

2.4.2 Sampling and evaluation of nonlinear constraints

For the purpose of constraint learning, we require data over variables and corresponding left-hand-side values of nonlinear constraints. The importance of the quality of data for the accuracy of machine learning tasks is well known and studied since the 1990's [25]. Thus, the distribution of data points used for constraint learning is critical. The samples over dom($\mathbf{x}$) should be sufficiently space-filling so that the behavior of each constraint is captured over the whole dom($\mathbf{x}$). In addition, we require sufficient concentration of points near the constraint boundary so that learners are adequately trained to predict the feasibility of near-feasible points.

To achieve both of these objectives, we take a disciplined approach to sampling, and generate data over dom($\mathbf{x}$) for each constraint in several stages. Note that the sampling and evaluation steps in the following subsections are performed constraintwise.

Boundary sampling

We first sample the corners of the $\mathbf{x}$ hypercube for the constraint, defined by $x_k \in [\underline{x}_k, \overline{x}_k],\ k \subseteq [n]$, in an effort to capture extremal points. We call this boundary sampling. This is combinatorial in the number of variables in each nonlinear constraint; a constraint with p bounded variables would require 2^p samples. In practice, we sample a limited combination of corner points, depending on the number of variables in the constraint.

Optimal Latin hypercube sampling

Next, we implement optimal Latin hypercube (OLH) sampling over the $\mathbf{x}$ hypercube. There is a wealth of literature starting with McKay et al. [67] that demonstrates the strength of Latin hypercube (LH) sampling versus other methods for DoE. However, LH sampling is not in general a *maximum entropy sampling scheme* [83], i.e. the samples from LHs do not optimize information gained about the underlying system. OLH sampling is the entropy maximizing variant of LH sampling for a uniform prior, where our entropy function is the pairwise Euclidian distances between sample points [3]. The uniform prior assumption is logical since we do not have or require an initial guess for where in the $\mathbf{x}$ hypercube the optimal solution will land, and the constraints are treated as black boxes.

OLH sampling, unlike standard LHs, is space-filling and thus useful for learning the global behavior of constraints using ML models. In practice, OLH generation is time-consuming and impractical. Instead, we use an efficient heuristic proposed by Bates et al. [4], which uses a permutation genetic algorithm to find near-optimal solutions to the OLH problem with low computational cost. We terminate the genetic algorithm prematurely in our optimization scheme, since samples are not required to be optimally distributed.

Constraint evaluation

We use the samples to either compute the left-hand-side of the constraint, or the feasibility of the constraint if the left-hand-side is not available. If the constraint is an equality $h_j(\mathbf{x}) = 0$, we relax it and treat it as an inequality $h_j(\mathbf{x}) \geq 0$ until Section 2.4.4. The result is a $\{0, 1\}^n$ feasibility vector corresponding to each of the n samples, defining the classes for the classification problem.

If desired, assuming that constraints use a common set of samples, it is possible to lump the feasibility of a set of inequality constraints by taking the row-wise minimum of their joint feasibility over the same data. This can reduce the model complexity, but we currently do not consider this in our method.

kNN quasi-Newton sampling

The previous sampling methods achieve a space-filling distribution of samples in $\mathrm{dom}(\mathbf{x})$ to enable approximating OCT-Hs to learn the feasibility of each constraint in a global sense. We still require sufficient concentration of points near the constraint boundary, i.e. points $\tilde{\mathbf{x}}_i$ so that $g(\tilde{\mathbf{x}}_i) \approx 0$, so our OCT-H models are trained to classify such near-feasible points accurately.

Assuming that the first stage sampling and evaluation has found at least one feasible point to the constraint, in this step, we attempt to sample near the constraint boundary using a method we've developed called kNN quasi-Newton sampling. The method hinges on using kNN to generate near-feasible neighborhoods for the constraint over previous data $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, and using approximate gradients in these neighborhoods to find new near-feasible samples $\tilde{\mathbf{u}}$, with vanishing $g(\tilde{u}_i) = \epsilon \to 0$. We present the method in Algorithm 1.

```
Algorithm 1: kNN quasi-Newton sampling
  Result: Sample points near the feasibility boundary of constraints.
  Find $k = p + 1$ nearest neighbors: $\xi = [k\text{NN}(\tilde{\mathbf{x}}_i,\ \tilde{\mathbf{x}}),\ \forall i \in [n]]$;
  Classify feasibility kNN patches: $\phi \in \{\text{feasible}, \text{infeasible}, \text{mixed}\}^{\text{n}}$;
  Initialize new sample container $\tilde{\mathbf{u}} = []$. ;
  for $i \in [n]$: do
      if $\phi_i$ = mixed and $\tilde{\mathbf{x}}_i$ infeasible then
          for $j \in [k]$ do
              if $\tilde{\mathbf{x}}_{\xi_{i,j}}$ feasible then
                  Augment $\tilde{\mathbf{u}}$: secant method$(\tilde{\mathbf{x}}_i,\ \tilde{y}_i,\ \tilde{\mathbf{x}}_{\xi_{i,j}},\ \tilde{y}_{\xi_{i,j}})$
```

The method is described as follows. Starting from space-filling data $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ where $\tilde{y}_i = g(\tilde{\mathbf{x}}_i)$, we find the k-nearest points for each sampled point $\tilde{\mathbf{x}}_i$ in the 0-1 normalized $\mathbf{x}$ hypercube. In our particular implementation, we use $k = p+1$, where p is the number of variables in constraint $g(\mathbf{x}) \geq 0$. For each kNN cluster with index i centered at $\tilde{\mathbf{x}}_i$ with $k-1$ neighbor indices ξ_i, we determine if all sample points are feasible, all points are infeasible, or points are mixed-feasibility.

In each cluster with mixed-feasibility points, we perform the secant method between points of opposing feasibility. The secant method is an approximate root finding algorithm defined by the following recurrence relation

$$\tilde{\mathbf{x}}_k = \tilde{\mathbf{x}}_j - \tilde{y}_j \frac{\tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i}{\tilde{y}_j - \tilde{y}_i}, \tag{2.4}$$

where $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ are points of opposing feasibility in the same mixed-feasibility neighborhood, and $\tilde{\mathbf{x}}_k$ is a new candidate root. The secant method thus allows us to efficiently generate roots $\tilde{\mathbf{x}}_k$ that would be expected to be near the constraint boundary, using combinations of points $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ from the space-filling OLH samples. We ensure that each pair of kNN-adjacent points on the constraint boundary results in only one new point, by only sampling within mixed-feasibility kNN cells if their centroid is infeasible, and then only sampling between the infeasible centroid and surrounding feasible points in the kNN cell.

Once we have performed the kNN sampling process and have new samples $\tilde{\mathbf{u}}$, we evaluate the left-hand-side $g(\mathbf{x})$ over the samples and add them to data $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ before proceeding to the tree training step.

2.4.3 Decision tree training

We use trees to approximate the nonlinear constraints in our global optimization problem due to their MIO representability, which we will demonstrate in Section 2.4.4. We use software from the company IAI in building, training and storing problem data in the form of OCT-Hs and ORT-Hs [48]. We train trees exclusively with hyperplane splits due to their higher approximation accuracy and lower tree complexity.

The trees are trained on all available data instead of a subset of the data as would be expected in traditional ML. In addition, we penalize tree complexity very little. This is because our data is noise-free, and approximation accuracy is important in the global optimization setting. In the case where the constraints are generated on noisy data, we would allow for the splitting of data into training and test sets, and cross-validate over a range of parameters.

We use the base OCT-H and ORT-H parameters in Table 2.1 within IAI when initializing constraint learning instances. These parameters are used for all computational benchmarks throughout the chapter unless stated otherwise. The parameters have been chosen to balance tree accuracy with tree complexity and associated computational cost, and may be tuned by users as they find necessary.

Parameter	OCT-H	ORT-H
Hyperplane sparsity	All	All
Regression sparsity	-	All
Max depth	5	5
Complexity factor	10^{-6}	10^{-6}
Minbucket	0.01	0.02
Random tree restarts	10	10
Hyperplane restarts	5	5

Table 2.1: Parameters for base decision trees in constraint learning.

Our training loss function for OCT-Hs is misclassification error. If a tree is a function that maps feature inputs into classes ($T : \mathbf{x} \to y$), the misclassification error is simply the weighted proportion of samples that are misclassified by the tree, where $\mathbb{I}$ is the indicator function and w_i are the sample weights. An exact classifier would have a misclassification error of 0.

$$\text{misclassification error} = \frac{1}{n} \frac{\sum_{i=1}^{n} w_i \cdot \mathbb{I}(T(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{n} w_i}.$$

For ORT-Hs used to approximate objective functions, we use $1 - \mathrm{R}^2$ as the loss function, where R^2 is the coefficient of determination. An exact regressor would have a $1 - \mathrm{R}^2$ value of 0.

$$1 - \mathrm{R}^2 = \frac{\sum_{i=1}^{n}(T(\mathbf{x}_i) - y_i)^2}{\sum_{i=1}^{n}(T(\mathbf{x}_i) - \bar{y})^2}, \text{ where } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

2.4.4 MI approximation

From this section forward, we recognize that the global optimization problem is approximated constraint-wise, and introduce indices $i \in I$ and $j \in J$ for the inequality and equality constraints respectively. Having classified the feasible space of nonlinear inequalities $g_i(\mathbf{x}) \geq 0,\ i \in I$ and relaxed nonlinear equalities $h_j(\mathbf{x}) \geq 0,\ j \in J$ using OCT-Hs, we retighten equalities to $h_j(\mathbf{x}) = 0,\ j \in J$, and pose the feasible $\mathbf{x}$-domains of each tree as unions of polyhedra. In this section, we define mathematically the set of disjunctive MI-linear constraints that represent the trees exactly.

Nonlinear inequalities

The tree T_i that classifies the feasible set of nonlinear inequality $g_i(\mathbf{x}) \geq 0$ has a set of leaves L_i, where a subset of leaves $L_{i,1} \subset L_i$ are classified feasible (where the indicator function $\mathbb{I}(g_i(\mathbf{x}) \geq 0) = 1$) and $L_{i,0} \subset L_i$ are classified infeasible ($\mathbb{I}(g_i(\mathbf{x}) \geq 0) = 0$). The decision path to each leaf defines a set of separating hyperplanes, $H_{i,l}$, where $H_{i,l,-}$ and $H_{i,l,+}$ are the set of leftward (less-than) and rightward (greater-than) splits required to reach leaf l respectively. The feasible polyhedron of tree T_i at feasible leaf $l \in L_{i,1}$ is thus defined as

$$\mathbf{P}_{i,l} = \{\mathbf{x} : \boldsymbol{\alpha}_h^\top \mathbf{x} \leq \beta_h,\ \forall\ h \in H_{i,l,-}\ ;\ \boldsymbol{\alpha}_h^\top \mathbf{x} \geq \beta_h,\ \forall\ h \in H_{i,l,+}\}. \tag{2.5}$$

The feasible set of $\mathbf{x}$ over constraint $g_i(\mathbf{x}) \geq 0$ is approximated by the union of the feasible polyhedra in (2.5). More formally,

$$\mathbf{x} \in \bigcup_{l \in L_{i,1}} \mathbf{P}_{i,l}. \tag{2.6}$$

This union-of-polyhedra representation can described by a set of disjunctive constraints involving a big-M formulation. Vielma [91] describes many such “projected" formulations; the specific disjunctive representation of OCT-Hs approximating nonlinear inequalities is as follows:

$$\mathbf{x} \in \bigcup_{l \in L_{i,1}} \mathbf{P}_{i,l} \iff \begin{cases} \{\ \boldsymbol{\alpha}_h^\top \mathbf{x} \leq \beta_h + M(1 - z_{i,l}),\ \forall\ h \in H_{i,l,-}\ ; \\ \ \beta_h \leq \boldsymbol{\alpha}_h^\top \mathbf{x} + M(1 - z_{i,l}),\ \forall\ h \in H_{i,l,+}\},\ \forall\ l \in L_{i,1}, \\ \displaystyle\sum_{l \in L_{i,1}} z_{i,l} = 1, \\ z_{i,l} \in \{0,1\},\ l \in L_{i,1}, \\ M > |\beta_h|,\ M > \max\limits_{\mathrm{dom}(\mathbf{x})} |\boldsymbol{\alpha}_h^\top \mathbf{x}|,\ \forall\ h \in H_{i,l},\ l \in L_{i,1}. \end{cases} \tag{2.7}$$

Membership of $\mathbf{x}$ in polyhedron $\mathbf{P}_{i,l}$ is defined by binary variable $z_{i,l}$. The constraint $\sum_{l \in L_{i,1}} z_{i,l} = 1$ ensures that $\mathbf{x}$ is in exactly one feasible polyhedron. However, the formulation above requires knowing the value of M with sufficient accuracy, which can be difficult in practice. The value of M is important; too small an M means that the constraint is insufficiently enforced, and too large an M can cause numerical issues. Knowing M to a sufficient tolerance can require solving the inner maximization in (2.7) over $\mathrm{dom}(\mathbf{x})$, and even declaring a separate M_h for each separating hyperplane $h \in H_{i,l}$.

Alternatively, we derive a representation that completely avoids the need to compute big-M values, since we restrict ourselves to $\mathbf{x} \in [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$. The tradeoff is that we require the addition of auxiliary variables $\mathbf{y}_l \in \mathbb{R}^{p_i}$, for each leaf $l \in L_{i,1}$, where p_i is the dimension of variables in constraint i. We present the big-M free representation of OCT-Hs used to approximate nonlinear inequalities in (2.8). The formulation is an application of basic extended disjunctive formulations for defining unions of polyhedra, as detailed by Vielma [91].

$$\mathbf{x} \in \bigcup_{l \in L_{i,1}} \mathbf{P}_{i,l} \iff \begin{cases} \{\ \boldsymbol{\alpha}_h^\top \mathbf{y}_l \leq \beta_h z_{i,l},\ \forall\ h \in H_{i,l,-}\ ; \\ \ \beta_h z_{i,l} \leq \boldsymbol{\alpha}_h^\top \mathbf{y}_l,\ \forall\ h \in H_{i,l,+}\}\ \forall\ l \in L_{i,1}, \\ \mathbf{y}_l \in [\underline{\mathbf{x}} z_{i,l}, \overline{\mathbf{x}} z_{i,l}],\ l \in L_{i,1}, \\ \displaystyle\sum_{l \in L_{i,1}} \mathbf{y}_l = \mathbf{x}, \\ \displaystyle\sum_{l \in L_{i,1}} z_{i,l} = 1, \\ z_{i,l} \in \{0,1\},\ l \in L_{i,1}. \end{cases} \tag{2.8}$$

Just as the big-M formulation, whether or not $\mathbf{x}$ lies in polyhedron $\mathbf{P}_{i,l}$ is defined by binary variable $z_{i,l} \in \{0,1\}$. If $\mathbf{x}$ is in $\mathbf{P}_{i,l}$, then $\mathbf{x} = \mathbf{y}_l$. If not, $\mathbf{y}_l = \mathbf{0}$. Thus $\mathbf{x}$ can only lie in the leaves of T_i that are classified feasible.

Notably, formulation (2.8) is *locally ideal*, i.e. its continuous relaxation has at least one basic feasible solution, and all its basic feasible solutions are integral in $\mathbf{z}_i$ [91]. This confers computational advantages in optimization over such disjunctions compared to its big-M variant. Since disjunctive formulation (2.8) is tractable and big-M free, we implement it in OCT-HaGOn.

Nonlinear equalities

Nonlinear equalities can also be approximated by OCT-Hs. To do so, we simply relax $h_j(\mathbf{x}) = 0$ to $h_j(\mathbf{x}) \geq 0$ and fit an OCT-H T_j to the feasible set of this constraint, with polyhedra $\mathbf{P}_{j,l}$, where l can lie in feasible leaves $L_{j,1}$ and infeasible leaves $L_{j,0}$. The feasible set of the original equality must be represented by the union of the polyhedral faces between the feasible and infeasible leaves. It is critical to note however that this is not equivalent to the union of polyhedral faces, $\mathbf{x} \in \bigcup_{l \in L_j} \text{faces}(\mathbf{P}_{j,l})$, since some of the faces separate two feasible spaces from each other, and thus would not be valid constraint boundaries. We are only interested in polyhedral faces that separate feasible polyhedra from infeasible polyhedra, where $h_j(\mathbf{x}) \geq 0$ and $h_j(\mathbf{x}) \leq 0$. Therefore the approximate equality is the union of intersections of all permutations of a feasible polyhedron with an infeasible polyhedron,

$$\mathbf{x} \in \bigcup_{l_0 \in L_{j,0},\ l_1 \in L_{j,1}} \{\mathbf{P}_{j,l_0} \cap \mathbf{P}_{j,l_1}\}. \tag{2.9}$$

To ensure that $\mathbf{x}$ lies on a face between a feasible and an infeasible polyhedron, we allocate a binary variable $z_{j,l}$ for each leaf $l \in L_j$. We make sure that $\mathbf{x}$ lies in exactly one feasible and one infeasible polyhedron by having exactly two non-zero $z_{j,l}$'s, one in a feasible leaf $l \in L_{j,1}$ and the other in an infeasible leaf $l \in L_{j,0}$. Thus we represent the approximate equality as the following set of disjunctive big-M constraints, where $L_j = \{L_{j,1} \cup L_{j,0}\}$ are the combined set of feasible and infeasible leaves of tree T_j.

$$\mathbf{x} \in \bigcup_{\substack{l_0 \in L_{j,0},\\ l_1 \in L_{j,1}}} \{\mathbf{P}_{j,l_0} \cap \mathbf{P}_{j,l_1}\} \iff \left\{ \begin{array}{l} \{\ \boldsymbol{\alpha}_h^\top \mathbf{x} \leq \beta_h + M(1 - z_{j,l}),\ \forall\ h \in H_{j,l,-}\ ; \\ \ \beta_h \leq \boldsymbol{\alpha}_h^\top \mathbf{x} + M(1 - z_{j,l}),\ \forall\ h \in H_{j,l,+}\},\ \forall l \in L_j, \\ \displaystyle\sum_{l \in L_{j,0}} z_{j,l} = 1, \quad \sum_{l \in L_{j,1}} z_{j,l} = 1, \\ z_{j,l} \in \{0,1\}, \quad l \in L_j. \end{array} \right. \tag{2.10}$$

This guarantees that $\mathbf{x}$ falls on a polyhedral face that separates a feasible and infeasible polyhedron, thus approximating $h_j(\mathbf{x}) = 0$. As we have done for nonlinear inequalities, we can come up with an equivalent big-M-free formulation as follows, and implement it in OCT-HaGOn.

$$\mathbf{x} \in \bigcup_{\substack{l_0 \in L_{j,0},\\ l_1 \in L_{j,1}}} \{\mathbf{P}_{j,l_0} \cap \mathbf{P}_{j,l_1}\} \iff \left\{ \begin{array}{l} \{\ \boldsymbol{\alpha}_h^\top \mathbf{y}_l \leq \beta_h z_{i,l},\ \forall\ h \in H_{i,l,-}\ ; \\ \ \beta_h z_{i,l} \leq \boldsymbol{\alpha}_h^\top \mathbf{y}_l,\ \forall\ h \in H_{i,l,+}\},\ \forall\ l \in L_j, \\ \mathbf{y}_l \in [\underline{\mathbf{x}} z_{i,l}, \overline{\mathbf{x}} z_{i,l}],\ l \in L_j, \\ \displaystyle\sum_{l \in L_{i,1}} \mathbf{y}_l = \mathbf{x}, \quad \sum_{l \in L_{i,0}} \mathbf{y}_l = \mathbf{x}, \\ \displaystyle\sum_{l \in L_{i,1}} z_{i,l} = 1, \quad \sum_{l \in L_{i,0}} z_{i,l} = 1, \\ z_{i,l} \in \{0,1\}, \quad l \in L_j. \end{array} \right. \tag{2.11}$$

Note that nonlinear equalities pose the greatest challenge for any global optimization method, since the ϵ-feasible space of equalities is restrictive.

Nonlinear objectives

We treat nonlinear objectives $f(\mathbf{x})$ differently than constraints. Constraints are represented well by classifiers because constraints partition the space of $\mathbf{x}$ into feasible and infeasible classes. Nonlinear objectives however are continuous with respect to $\mathbf{x}$, and are thus better approximated by regressors. To approximate a nonlinear objective function $f(\mathbf{x})$, we train an ORT-H on sample data $\{\tilde{\mathbf{x}}_i, f(\tilde{\mathbf{x}}_i)\}_{i=1}^n$, and replace the nonlinear objective with the auxiliary variable f^*. We lower bound the value of f^* using the disjunctive constraints derived from the ORT-H, thus approximating the original objective function.

We can apply the same logic to constraints of the form $\mathbf{a}^\top \mathbf{x} + b \geq g(\mathbf{x})$, where the left-hand-side is affine and separable from the nonlinear component $g(\mathbf{x})$. Since $\mathbf{a}^\top \mathbf{x} + b$ is linear and MIO-compatible, we instead train an ORT-H on sample data $\{\tilde{\mathbf{x}}_i, g(\tilde{\mathbf{x}}_i)\}_{i=1}^n$, and make sure that $\mathbf{a}^\top \mathbf{x} + b$ is lower bounded by the approximating ORT-H. It is the choice of the user whether or not to use OCT-Hs or ORT-Hs to approximate separable constraints, but in general an ORT-H is more accurate in these cases. All problems addressed in Section 2.6 treat constraints as non-separable, and use classifiers to approximate them. To solve the satellite scheduling problem in Section 2.7.2, we take advantage of this separability and choose to train ORT-Hs instead.

Since an ORT-H is an OCT-H with additional regressors added to each leaf, the disjunctive constraints in (2.8) and (2.11) apply with minor modifications described as follows. L_f is the set of leaves of the approximating ORT-H; assuming that $f(\mathbf{x})$ can be evaluated on $\text{dom}(\mathbf{x})$, all leaves $l \in L_f$ of the ORT-H can feasibly contain $\mathbf{x}$, meaning that the disjunctions are applied to all leaves instead of a subset of the leaves of the tree. Each leaf $l \in L_f$ has a set of separating hyperplanes that is described by its decision path, as well as an additional separating hyperplane described by the regressor in each leaf.

For objectives and separable inequalities, instead of using the regressor within each leaf of the ORT-H directly, we run a secondary linear regression problem on the points within each leaf to find the tightest lower bounding hyperplane on the data. This allows us to have an approximate relaxation of the constraint or objective function, and tighten the relaxation later via solution repair in Section 2.4.6.

2.4.5 Solution of MIO approximation

Having represented the feasible space of inequality and equality constraints as a unions of polyhedra, we have the following final problem.

$$\begin{aligned}
\min_{x} \quad & f^* \\
\text{s.t.} \quad & f^*, \mathbf{x} \in \bigcup_{l \in L_f} \mathbf{P}_{i,l}, \\
& \mathbf{x} \in \bigcup_{l \in L_{i,1}} \mathbf{P}_{i,l}, \ \forall\, i \in I, \\
& \mathbf{x} \in \bigcup_{l_0 \in L_{j,0},\ l_1 \in L_{j,1}} \{\mathbf{P}_{j,l_0} \cap \mathbf{P}_{j,l_1}\}, \ \forall j \in J, \\
& \mathbf{A}\mathbf{x} \geq \mathbf{b}, \ \mathbf{C}\mathbf{x} = \mathbf{d}, \\
& x_k \in [\underline{x}_k, \overline{x}_k], \ k \in [n].
\end{aligned} \tag{2.12}$$

This is a MILO that can be efficiently solved using branch-and-bound methods. We use CPLEX for this purpose, since it is available free of charge to solve small scale MILO instances.

2.4.6 Solution checking and repair

The optimum obtained in Section 2.4.5 is likely to be near-optimal and near-feasible to the original global optimization problem, since the MIO is approximate. To repair the solution in case of suboptimality or infeasibility, we devise and present a local search procedure based on projected gradient descent (PGD). PGD is a method for constrained gradient descent that is reliable, scalable and fast for the local optimization required to restore feasibility and optimality to approximate solutions. It relies on using gradients of the constraints and objective to simultaneously reduce constraint violation (by projecting $\mathbf{x}^*$ onto the feasible space of $\mathbf{x}$) and the objective function value. Our particular implementation of PGD solves a series of gradient-driven MIO problems to do so.

To obtain the gradients of explicit and inexplicit constraints, we leverage automatic differentiation (AD), and specifically *forward mode AD*. Forward mode AD looks at the fundamental mathematical operations involved in evaluating the constraint functions, and thus computes the gradient of each constraint exactly at any solution $\mathbf{x}^*$ [90]. Unlike finite differentiation, AD does not require additional function evaluations or discretization, and unlike symbolic differentiation, it doesn't require the constraints to be explicit.

The proposed PGD method begins by first evaluating the objective and all constraints at $\mathbf{x}^*$, the last known optimum, as well as their gradients. The disjunctive approximations of nonlinear inequality constraints are replaced by linear approximators based on the local constraint gradient, depending on the feasibility of each constraint:

$$g_i(\mathbf{x}) \geq 0 \rightarrow \begin{cases} \nabla g_i(\mathbf{x}^*)^\top \mathbf{d} + g_i(\mathbf{x}^*) \geq 0, \text{ if } g_i(\mathbf{x}^*) \geq 0, \\ \nabla g_i(\mathbf{x}^*)^\top \mathbf{d} + g_i(\mathbf{x}^*) + \lambda_i \geq 0, \text{ if } g_i(\mathbf{x}^*) \leq 0, \end{cases} \tag{2.13}$$

where $\mathbf{d} \in \mathbb{R}^n$ is the descent direction, and $\lambda_i \in \mathbb{R}^+$ is an inequality relaxation variable. Similarly, we replace the MI approximations of equalities with their local linear approximators, but always include relaxation variables regardless of the level of infeasibility of the constraints, as shown in (2.14).

$$h_j(\mathbf{x}) = 0 \rightarrow \begin{cases} \nabla h_j(\mathbf{x}^*)^\top \mathbf{d} + h_j(\mathbf{x}^*) + \mu_j \geq 0, \\ \nabla h_j(\mathbf{x}^*)^\top \mathbf{d} + h_j(\mathbf{x}^*) \leq \mu_j, \end{cases} \tag{2.14}$$

where $\mu_j \in \mathbb{R}^+$ is an equality relaxation variable. This relaxation is for two reasons. The first is that, in presence of equalities, the local PGD step may be infeasible due to conflicting equality constraints. The second is that each PGD step will involve solving a quadratic program, which can only be solved to given numerical precision. This precision, while low, is non-zero.

Thus we introduce a constraint tightness tolerance parameter ϕ, and say that an inequality $g_i(\mathbf{x}) \geq 0$ is feasible at $\mathbf{x}^*$ if $g_i(\mathbf{x}^*) \geq -\phi$. If all inequality constraints are feasible to tolerance, relaxation variables λ are only required on the inequalities where $0 \geq g_i(\mathbf{x}^*) \geq -\phi,\ i \in I$, by the condition in (2.13). In that case, we perform a simple gradient descent step. This involves solving the quadratic optimization problem in (2.15), where γ is the infeasibility penalty coefficient, α is the step size within a 0-1 normalized $\mathbf{x}$ hypercube, r is the step size decay rate, t is the current PGD iteration and T is the maximum number of iterations.

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{d},\lambda,\mu} \quad & \nabla f(\mathbf{x}^*)^\top \mathbf{d} + \gamma(||\lambda||_2^2 + ||\mu||_2^2) \\
\text{s.t.} \quad & \mathbf{x} = \mathbf{x}^* + \mathbf{d}, \\
& \left|\left|\frac{\mathbf{d}}{\overline{\mathbf{x}} - \underline{\mathbf{x}}}\right|\right|_2^2 \le \alpha \exp\Big(\frac{-\mathrm{rt}}{\mathrm{T}}\Big), \\
& \left\{\begin{array}{ll} \nabla g_i(\mathbf{x}^*)^\top \mathbf{d} + g_i(\mathbf{x}^*) \ge 0, & \text{if } g_i(\mathbf{x}^*) \ge 0 \\ \nabla g_i(\mathbf{x}^*)^\top \mathbf{d} + g_i(\mathbf{x}^*) + \lambda_i \ge 0, & \text{if } -\phi \le g_i(\mathbf{x}^*) \le 0 \end{array}\right\}, \ \forall i \in I, \\
& \left\{\begin{array}{l} \nabla h_j(\mathbf{x}^*)^\top \mathbf{d} + h_j(\mathbf{x}^*) + \mu_j \ge 0, \\ \nabla h_j(\mathbf{x}^*)^\top \mathbf{d} + h_j(\mathbf{x}^*) \le \mu_j, \end{array}\right\}, \ \forall j \in J, \\
& \mathbf{A}\mathbf{x} \ge \mathbf{b}, \ \mathbf{C}\mathbf{x} = \mathbf{d}, \\
& x_k \in [\underline{x}_k, \overline{x}_k], \ \forall k \in [n] \\
& \left\{\begin{array}{ll} \lambda_i = 0, & \text{if } g_i(\mathbf{x}^*) \ge 0 \\ \lambda_i \ge 0, & \text{if } g_i(\mathbf{x}^*) \le 0 \end{array}\right\}, \ \forall i \in I, \\
& \mu_i \in \mathbb{R}_+, \ j \in J.
\end{aligned}
\tag{2.15}
$$

We exponentially decrease the allowed step size **d** as defined in (2.15), to aid convergence and break cycles that may result.

If the current solution $\mathbf{x}^*$ is infeasible beyond tolerance to any constraints, we take a projection-and-descent step. This modifies the objective and first two constraints in (2.15) by removing the step size constraint on **d**, and augmenting the objective function with a projection distance penalty with β as a parameter, as shown in (2.16):

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{d},\lambda,\mu} \quad & \nabla f(\mathbf{x}^*)^\top \mathbf{d} + \beta \left|\left|\frac{\mathbf{d}}{\overline{\mathbf{x}} - \underline{\mathbf{x}}}\right|\right|_2^2 + \gamma(||\lambda||_2^2 + ||\mu||_2^2) \\
\text{s.t.} \quad & \mathbf{x} = \mathbf{x}^* + \mathbf{d}, \\
& \vdots
\end{aligned}
\tag{2.16}
$$

This quadratic optimization problem approximates the closest feasible projection of **x** onto the feasible space of nonlinear constraints.

The gradient and projected gradient steps defined above require knowing the maximum range on all variables, $\overline{\mathbf{x}} - \underline{\mathbf{x}}$. If this range is not provided for variable x_k, then we assume $\overline{x}_k - \underline{x}_k = \max(\overline{\mathbf{x}}) - \min(\underline{\mathbf{x}})$. The convergence of the PGD is much stronger with user-provided bounds however. We repeat the above PGD steps on new incumbent solutions until the final two solutions are feasible to all constraints, and the improvement in original objective function $f(\mathbf{x})$ is less than absolute tolerance ϵ.

The PGD algorithm introduces many parameters, whose default values are defined in Table 2.2. While this adds additional complexity to the solution procedure, the descent procedure is intuitive to tune, and the current implementation warns the user in case parameters require examination. In addition, the parameters are applied to 0-1 normalized quantities over the **x** hypercube wherever possible. For all examples in this chapter, the default PGD parameters from Table 2.2 apply unless stated otherwise.

Parameter	Description	Value
γ	Infeasibility penalty	10^6
β	Step penalty	10^4
α	Step size	10^{-3}
r	Decay rate	2
T	Maximum iterations	100
ϵ	Absolute tolerance	10^{-4}
ϕ	Tightness tolerance	10^{-8}

Table 2.2: Parameters for PGD repair procedure.

2.5 Demonstrative example

Consider the following modified mixed-integer nonlinear optimization problem from Duran and Grossmann [34]. For demonstrative purposes, the original nonlinear objective has been replaced with a linear objective, and variables $\mathbf{y}$ have been concatenated to $\mathbf{x}$ for consistency of notation.

$$\begin{aligned}
\min\ & f(\mathbf{x}) = 10x_1 - 17x_3 - 5x_4 + 6x_5 + 8x_6 \\
\text{s.t. } & g_1(\mathbf{x}) = 0.8\mathrm{log}(\mathrm{x}_2+1) + 0.96\mathrm{log}(\mathrm{x}_1 - \mathrm{x}_2 + 1) - 0.8\mathrm{x}_3 \geq 0, \\
& g_2(\mathbf{x}) = \mathrm{log}(\mathrm{x}_2+1) + 1.2\mathrm{log}(\mathrm{x}_1 - \mathrm{x}_2 + 1) - \mathrm{x}_3 - 2\mathrm{x}_6 + 2 \geq 0, \\
& x_1 - x_2 \geq 0, \quad 2x_4 - x_2 \geq 0, \\
& 2x_5 - x_1 + x_2 \geq 0, \quad 1 - x_4 - x_5 \geq 0, \\
& 0 \leq x_1 \leq 2,\ 0 \leq x_2 \leq 2,\ 0 \leq x_3 \leq 1, \\
& x_4, x_5, x_6 \in \{0,1\}^3.
\end{aligned} \tag{2.17}$$

We will focus on the nonlinear inequalities $g_1(\mathbf{x}) \geq 0$ and $g_2(\mathbf{x}) \geq 0$ as we implement the method step by step.

2.5.1 Standard form problem

Most global optimization problems are compatible with the standard form in Section 2.4.1 by construction. We demonstrate this by partitioning the original problem (2.17) below.

$\min\ f(\mathbf{x}) = 10x_1 - 17x_3 - 5x_4 + 6x_5 + 8x_6$	Objective
$\text{s.t.}\ g_1(\mathbf{x}) = 0.8\text{log}(\text{x}_2+1) + 0.96\text{log}(\text{x}_1 - \text{x}_2 + 1) - 0.8\text{x}_3 \geq 0,$	Nonlinear
$g_2(\mathbf{x}) = \text{log}(\text{x}_2+1) + 1.2\text{log}(\text{x}_1 - \text{x}_2 + 1) - \text{x}_3 - 2\text{x}_6 + 2 \geq 0,$	constraints
$x_1 - x_2 \geq 0, \quad 2x_4 - x_2 \geq 0,$	Linear
$2x_5 - x_1 + x_2 \geq 0, \quad 1 - x_4 - x_5 \geq 0,$	constraints
$0 \leq x_1 \leq 2,\ 0 \leq x_2 \leq 2,\ 0 \leq x_3 \leq 1,$	Variables
$x_4, x_5, x_6 \in \{0,1\}^3.$	and bounds

We pass the linear constraints, variables and bounds directly to the MIO model, and confirm that all variables in nonlinear constraints, in this case x_1, x_2, x_3 and x_6, are bounded. Note the presence of binary x_4, x_5 and x_6 in the problem as well.

2.5.2 Sampling and evaluation of nonlinear constraints

Next we generate samples over the nonlinear constraints using the procedure in Section 2.4.2. Note that $g_1(\mathbf{x}) \geq 0$ and $g_2(\mathbf{x}) \geq 0$ have 3 and 4 active variables, so samples are generated in $\mathbb{R}^3$ and $\mathbb{R}^4$ respectively. The resulting samples over $g_1(\mathbf{x}) \geq 0$ and their feasibilities are shown in Figure 2-4. Note that the samples span the whole $\mathbf{x}$ hypercube, but that there are certain concentrations of points, thanks to the kNN sampling procedure, that approximate the constraint boundary. This improves the ability of the approximating OCT-H to be both globally and locally accurate.

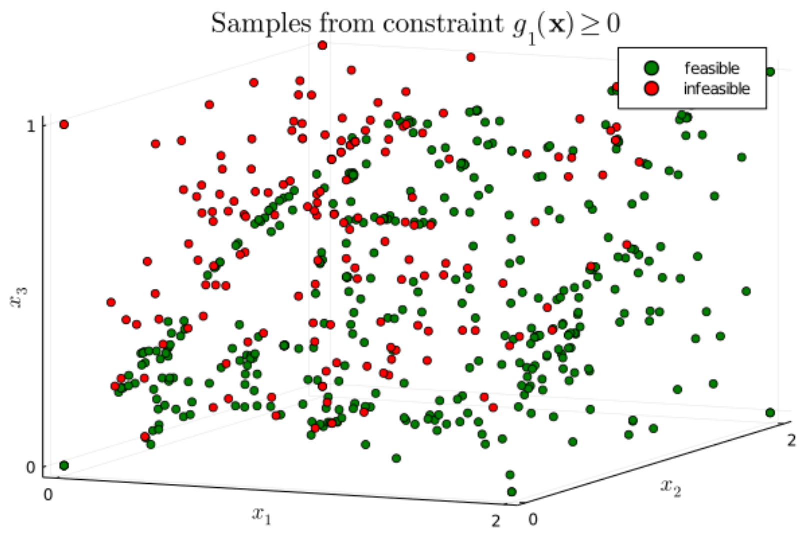
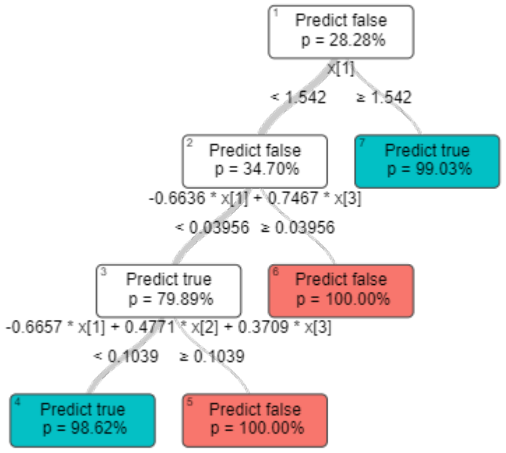


Figure 2-4: The distribution of data for constraint $g_1(\mathbf{x}) \geq 0$, generated by sampling procedures defined in Section 2.4.2.

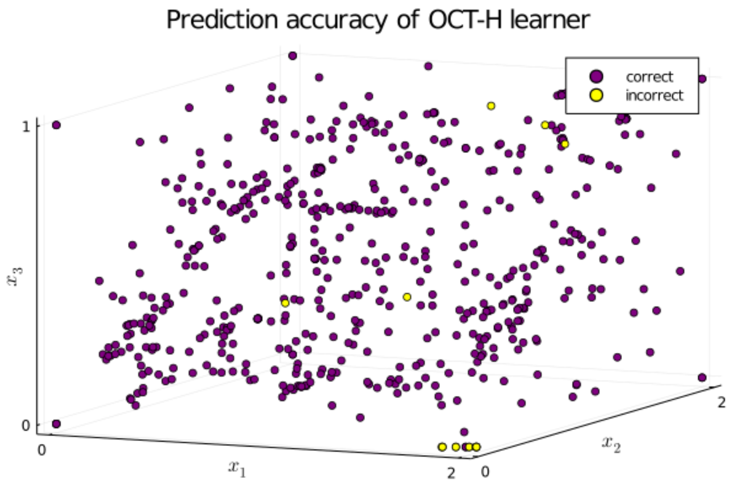
2.5.3 Decision tree training

We train two OCT-Hs to classify the feasible space of constraints $g_1(\mathbf{x}) \geq 0$ and $g_2(\mathbf{x}) \geq 0$. For demonstrative purposes, the trees were limited to a maximum depth of 3, as opposed to the standard depth of 5 used in OCT-HaGOn as defined in Table 2.1. The approximating OCT-H for $g_1(\mathbf{x}) \geq 0$ and the accuracy of its predictions are presented in Figure 2-5. Notably, the OCT-H approximator achieves a high degree of accuracy (97%) throughout $\text{dom}(\mathbf{x})$ with only two feasible leaves.



1 Predict false p = 28.28%
x[1]
< 1.542
≥ 1.542
2 Predict false p = 34.70%
7 Predict true p = 99.03%
-0.6636 * x[1] + 0.7467 * x[3]
< 0.03956
≥ 0.03956
3 Predict true p = 79.89%
6 Predict false p = 100.00%
-0.6657 * x[1] + 0.4771 * x[2] + 0.3709 * x[3]
< 0.1039
≥ 0.1039
4 Predict true p = 98.62%
5 Predict false p = 100.00%

(a) OCT-H over constraint $g_1(\mathbf{x}) \geq 0$ has two feasible and two infeasible leaves, and a depth of 3.

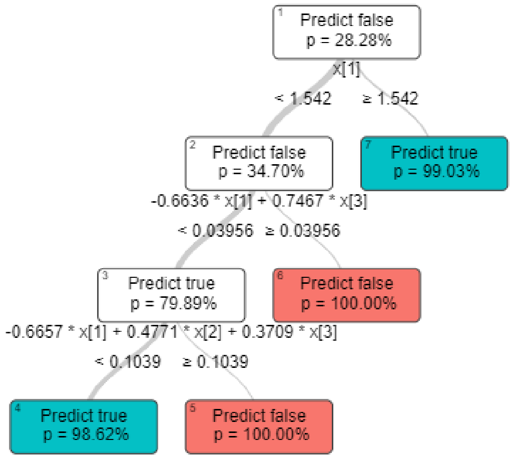


(b) OCT-H approximation is 97% accurate over 554 samples.

Figure 2-5: The approximating OCT-H achieves a high degree of accuracy, capturing both the global and local behavior of the constraint $g_1(\mathbf{x}) \geq 0$.

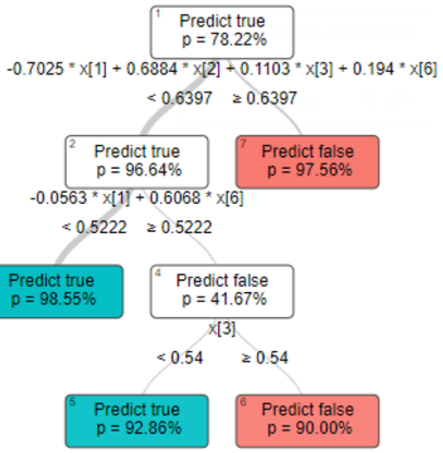
2.5.4 MI approximation

We pose the trees in a MIO-compatible form. As a bookkeeping note, auxiliary variables are introduced with two indices, the first indicating the constraint index, and the second indicating the numerical index of the leaf of the approximating OCT-H. This is consistent with the formulation in Section 2.4.4.



$$\begin{aligned}
[1,0,0]\cdot \mathbf{y}_{1,7} &\geq 1.542 z_{1,7},\\
[1,0,0]\cdot \mathbf{y}_{1,4} &\leq 1.542 z_{1,4},\\
[-0.6636, 0, 0.7467]\cdot \mathbf{y}_{1,4} &\leq 0.03956 z_{1,4},\\
[-0.6657, 0.4771, 0.3709]\cdot \mathbf{y}_{1,4} &\leq 0.1039 z_{1,4},\\
\mathbf{y}_{1,4} + \mathbf{y}_{1,7} = [x_1, x_2, x_3],&\ \ z_{1,4} + z_{1,7} = 1,\\
[0,0,0] z_{1,4} \leq \mathbf{y}_{1,4} &\leq [2,2,1] z_{1,4},\\
[0,0,0] z_{1,7} \leq \mathbf{y}_{1,7} &\leq [2,2,1] z_{1,7},\\
z_{1,4}, z_{1,7} &\in \{0,1\}^2.
\end{aligned} \tag{2.18}$$

Figure 2-6: $g_1(\mathbf{x}) \geq 0$ is approximated via 6 continuous and 2 binary auxiliary variables, and 6 linear constraints.



$$\begin{aligned}
[-0.7025, 0.6884, 0.1103, 0.194]\cdot \mathbf{y}_{2,3} &\leq 0.6397 z_{2,3},\\
[-0.0563, 0, 0, 0.6068]\cdot \mathbf{y}_{2,3} &\leq 0.5222 z_{2,3},\\
[-0.7025, 0.6884, 0.1103, 0.194]\cdot \mathbf{y}_{2,5} &\leq 0.6397 z_{2,5},\\
[-0.0563, 0, 0, 0.6068]\cdot \mathbf{y}_{2,5} &\geq 0.5222 z_{2,5},\\
[0,0,1,0]\cdot \mathbf{y}_{2,5} &\leq 0.54 z_{2,5},\\
\mathbf{y}_{2,3} + \mathbf{y}_{2,5} = [x_1, x_2, x_3, x_6],&\ \ z_{2,3} + z_{2,5} = 1,\\
[0,0,0,0] z_{2,3} \leq \mathbf{y}_{2,3} &\leq [2,2,1,1] z_{2,3},\\
[0,0,0,0] z_{2,5} \leq \mathbf{y}_{2,5} &\leq [2,2,1,1] z_{2,5},\\
z_{2,3}, z_{2,5} &\in \{0,1\}^2.
\end{aligned} \tag{2.19}$$

Figure 2-7: $g_2(\mathbf{x}) \geq 0$ is approximated via 8 continuous and 2 binary auxiliary variables, and 7 linear constraints.

Figure 2-6 shows the approximating tree for constraint $g_1(\mathbf{x}) \geq 0$, as well as its disjunctive representation as defined by (2.8). Since the constraint has three active variables $[x_1, x_2, x_3]$, and the tree has two feasible leaves with node indices 4 and 7, the disjunctive representation requires the definition of 6 auxiliary continuous variables $\mathbf{y}_{1,4} \in \mathbb{R}^3$ and $\mathbf{y}_{1,7} \in \mathbb{R}^3$, and two binary variables $z_{1,4}$ and $z_{1,7}$. The number of linear constraints required is 6, which is equal to the sum of the depths of each feasible leaf, plus 2 additional constraints defining the disjunctions.

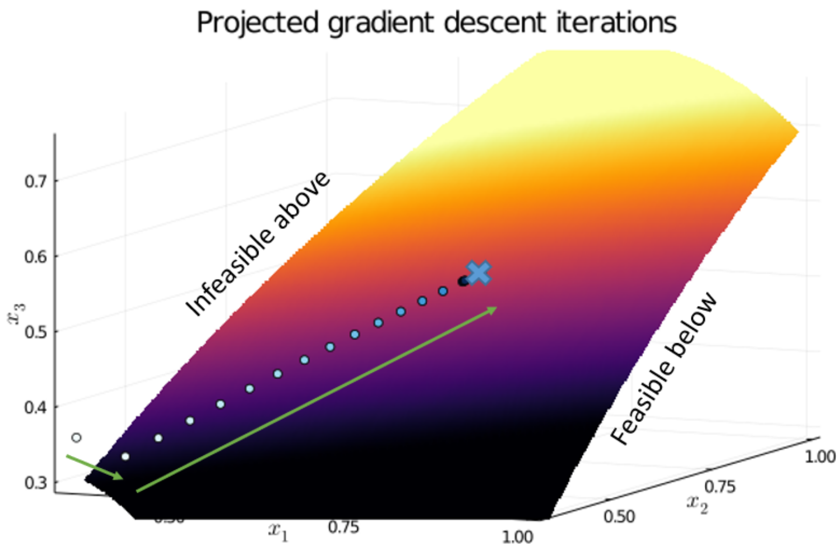
We approximate $g_2(\mathbf{x}) \geq 0$ in Figure 2-7, with four active variables $[x_1, x_2, x_3, x_6]$, using the same approach.

2.5.5 Solution of MIO approximation

As described Section 2.4.5, once the intractable constraints $g_1(\mathbf{x}) \geq 0$ and $g_2(\mathbf{x}) \geq 0$ are replaced with their tractable disjunctive approximations (2.18) and (2.19), the problem turns into a MILO that is tractable using commercial solvers. We solve the problem via CPLEX, and obtain a near-feasible, near-optimal solution with the objective value of -7.685 in Table 2-8a.

	x_1	x_2	x_3	x_4	x_5	x_6	$f(\mathbf{x})$
MIO	0.375	0.375	0.379	1.0	0.0	0.0	-7.685
PGD-repaired	0.699	0.699	0.530	1.0	0.0	0.0	-7.021

(a) The optimal solutions to demonstrative problem, pre- and post-PGD repair.



(b) The progress of the PGD method on the demonstrative example, plotted with respect to x_1, x_2 and x_3 on the surface of $g_1(\mathbf{x}) \geq 0$.

Figure 2-8: The MIO solution to the demonstrative example is successfully repaired to be feasible and locally optimal by the PGD method.

2.5.6 Solution checking and repair

We check whether the approximate solution $\mathbf{x}^*$ is feasible to the original optimization problem (2.17) by evaluating the two nonlinear constraints. Since constraint $g_1(\mathbf{x}) \geq 0$ is violated, we initiate the PGD repair procedure from Section 2.4.6. To do so, OCT-HaGOn replaces the MIO approximations of intractable constraints with the auto-differentiated gradients of the constraints at $\mathbf{x}^*$, and takes a local step to close the feasibility gap while descending along the objective. This is done iteratively, evaluating the objective function and nonlinear constraints at each step, until all constraints are feasible, and the change in the objective value falls below an absolute tolerance (10^{-4}). The path of the PGD algorithm is shown in Figure 2-8b, on the surface of constraint $g_1(\mathbf{x}) \geq 0$, which divides the feasible space of $\mathbf{x}$ in two. Note that this surface is unknown by the method, so it is remarkable that it projects towards it with remarkable accuracy in its first step, and then moves along the surface in a series of descent steps.

For this problem, the absolute tolerance of 10^{-4} was too small to converge definitively, so the PGD algorithm terminates at its maximum of 100 iterations, with the optimal objective value of $f(\mathbf{x}^*) = -7.021$ and the optimal solution in Table 2-8a.

2.6 Computational experiments on benchmarks

We apply OCT-HaGOn to a number of optimization problems from the literature, and benchmark it against other global optimizers. The software implementation of OCT-HaGOn can be found via the link in Appendix A.1. For the full list of optimizers used and their capabilities, please refer to Appendix A.2. We lead this section with a caveat. Since our approach is approximate, different random restarts of the solution procedure may yield different optima. However, experience implementing the method suggests that the method is consistent in finding the same optimum in most cases, and that random restarts reliably mitigate issues resulting from finding near-optimal solutions.

We first apply our method to five small benchmark problems from MINLPLib [21], and compare our results to those of BARON [82], a popular and effective commercial mixed integer nonlinear program (MINLP) solver. The types and numbers of constraints in the benchmarks are listed in Table 2.3. The results are shown in Table 2.4.

OCT-HaGOn is able to find the global optima for all five small benchmarks, matching the BARON solutions. OCT-HaGOn takes significantly longer to solve the small benchmarks than BARON. This is expected, since these problems have explicit constraints that only contain mathematical primitives BARON supports. Tree training time makes up the vast majority of the solution times for the small benchmarks; the MIO and PGD solution steps are efficient, taking less that 5% of the total time for each benchmark. Within the context of using optimization in design, where the optimization would be run many times to obtain a number of solutions on the Pareto frontier, OCT-HaGOn is competitive and even faster than BARON, since the MIO and PGD steps are solved in a small fraction of the time it takes for the BARON solver to solve a single instance of each MINLP.

We proceed by considering a set of six larger benchmarks from MINLPLib [21], as shown in Table 2.5. We also address the optimization problems using three commercially available solvers, IPOPT, CONOPT and BARON. Given the increased difficulty of these larger bench-

Problem Name	Continuous Variables	Integer Variables	Linear Constraints	Nonlinear Inequalities	Nonlinear Equalities	Nonlinear Objective
`minlp`	3	1	4	2	0	Y
`pool1`	7	0	2	4	0	N
`nlp1`	2	0	0	1	0	N
`nlp2`	3	0	0	0	3	N
`nlp3`	10	0	3	1	3	Y

Table 2.3: The five small nonlinear benchmarks from MINLPLib have a combination of nonlinear inequalities, equalities and objective.

Problem name	Objective		Time (s)		Solution	
	BARON	OCT-HaGOn	BARON	OCT-HaGOn	BARON	OCT-HaGOn
`minlp`	6.0098	6.0098	0.120	29.9	[0,1,0,1.3,0,1]	[0,1,0,1.3,0,1]
`pool1`	23.0	23.0	0.082	3.90	[4.0, 3.0, 1.0, 4.0, 0.0 2.12, 0.0]	[4.0, 3.0, 1.0, 4.0, 0.0 6.63, 0.0]
`nlp1`	-6.667	-6.667	0.106	0.461	[6, 0.667]	[6, 0.667]
`nlp2`	201.16	201.16	0.092	2.75	[6.29, 3.82, 201.16]	[6.29, 3.82, 201.16]
`nlp3`	-1161.34	-1161.34	1.265	17.7	[...]	[...]

Table 2.4: Solutions to the small benchmarks using OCT-HaGOn and BARON.

Problem name	Continuous Variables	Integer Variables	Linear Constraints	Nonlinear Inequalities	Nonlinear Equalities	Nonlinear Objective
`himmel16`	19	0	1	15	6	N
`kall_circles_c6b`	18	0	54	21	1	N
`pointpack08`	17	0	41	28	0	N
`flay05m`	23	40	61	5	0	N
`fo9`	111	72	326	18	0	N
`o9_ar4_1`	109	72	418	18	0	N

Table 2.5: The six larger benchmarks from MINLPLib. Note that the objective functions are linear in $\mathbf{x}$, and that nonlinearities are instead embedded in the constraints.

Problem name	Objective			Time (s)		GO
	GO	OCT-HaGOn	BK	Global	OCT-HaGOn	
`himmel16`	-0.6798	-0.8660^*	-0.8660	0.055	109.575	CONOPT
`kall_circles_c6b`	2.8104	2.1583*	1.9736	0.355	38.503	IPOPT
`pointpack08`	-0.2574	-0.2500	-0.2679	13.483	91.805	IPOPT
`flay05m`	64.498	64.499	64.498	0.212	9.515	CONOPT
`fo9`	23.464	23.464	23.464	959.090	29.534	BARON
`o9_ar4_1`	236.138	236.138	236.138	2283.281	1255.598	BARON

Table 2.6: Solutions to the larger benchmarks using commercial global optimizers (GOs) and OCT-HaGOn, against best known (BK) solutions.

marks, we allow OCT-HaGOn to use trees with maximum depth of 8, and generate double the number of samples per constraint compared to the small benchmarks.

The results are shown in Table 2.6, compared with the best known solutions as documented in MINLPLib. OCT-HaGOn finds the best known global optima for 4 out of 6 instances, and high performing solutions otherwise. Some modifications were required for a subset of the problems to be able to apply our method. The problems marked with an asterisk required the following changes to the algorithm:

- The `himmel16` test case contains a number of variables in nonlinear constraints that are unbounded. Using our little knowledge of the problem, we were able to make it compatible with our method by imposing bounds on all variables, $\mathbf{x} \in [-1, 1]^{19}$.

- The `kall_circles_c6b` example required increasing the step penalty and equality penalty to 10^8, to damp the PGD projection rate in order to avoid a conservative local optimum.

While these results are promising in showing that the method can scale to larger problems, they point to some practical considerations. The results show weak correlation between solution time and size of the problems; this is because the number of variables and complexity of nonlinearities in the approximated constraints tend to drive tree training time and thus total solution time. Additionally, as the problem size increases, it is not obvious whether tree training or MIO steps drive computational time, especially in the presence of integer variables. For the `himmel16` example, tree training takes 104 seconds of the 110 second total time, whereas for the `o9_ar4_1` benchmark, optimization time dominates, with training only taking 3 seconds out of nearly 21 minutes of total time. And while `fo9` and `o9_ar4_1` are of similar sizes and have similar constraints (both contain nonlinearities with inverses), they have dramatically different solution times.

2.7 Real world examples

In addition to the benchmarks, we test our method on two aerospace problems of varying complexity. We first solve a benchmark from the engineering literature, to show that the method can address real-world problems. We then apply OCT-HaGOn to a satellite on-orbit servicing problem that cannot be addressed using other global optimizers.

2.7.1 Speed reducer problem

The speed reducer problem is a nonlinear optimization problem posed in [41]. The problem aims to design a gearbox for an aircraft engine, subject to 11 specifications, geometry, structural and manufacturability constraints, in addition to variable bounds over $\mathbf{x} \in \mathbb{R}^7$. We apply our method to the problem as written in Appendix A.3 in standard form.

In Table 2.7, we compare different solutions to the speed reducer problem. Both OCT-HaGOn and IPOPT beat the best known optimum from [58]. In addition, OCT-HaGOn allows us to achieve all constraints with zero error after 4 iterations of the PGD algorithm

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Objective	Time (s)	Error
BK	3.5	0.7	17	7.3	7.7153	3.3503	5.2867	2994.472	476	10^{-6}
OCT-HaGOn	3.5	0.7	17	7.3	7.7153	3.3502	5.2867	2994.355	32.6	0
IPOPT	3.5	0.7	17.0*	7.3	7.7153	3.3502	5.2867	2994.355	4.2	10^{-7}

Table 2.7: Both OCT-HaGOn and IPOPT beat the best known (BK) solution of the speed reducer problem. In addition, OCT-HaGOn has 0 error on constraint satisfaction.

as shown in Appendix A.3, while the other two methods have small but nonzero error tolerances.

IPOPT was able to solve this particular nonlinear program (NLP) in 4.2 seconds, significantly faster than OCT-HaGOn, which took 32.6 seconds. However, this required a relaxation of the integrality of x_3. For this particular problem, this was not concerning since x_3 was lower bounded by its optimal value of 17. However, IPOPT cannot in general be used to solve MINLPs.

On a practical note, we would like to note the different levels of complexity in the OCT-H approximations of the underlying nonlinear constraints. Some constraints, while they look quite complex, have low-complexity tree approximators. Consider the following constraint $g_5(\mathbf{x}) \geq 0$ and its associated OCT-H approximator. The OCT-H model has a

$$g_5(\mathbf{x}) = 110x_6^3 - \left[\left(745\frac{x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right]^{0.5} \geq 0$$

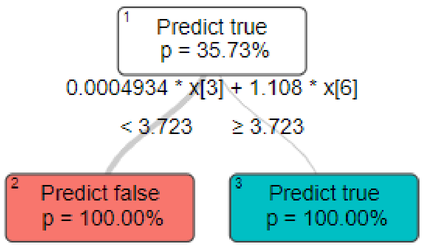


Figure 2-9: The constraint $g_5(\mathbf{x}) \geq 0$ is accurately approximated by a single separating hyperplane over dom($\mathbf{x}$).

single hyperplane that is able to approximate the function in the relevant dom($\mathbf{x}$) with perfect accuracy over 613 samples, as shown in Figure 2-9. Within the bounded dom($\mathbf{x}$), the nonlinear constraint is thus simplified to a linear constraint.

However, not all constraints are straightforward to represent via unions of polyhedra. Consider the objective function, which is a 5th order polynomial (2.20). In this particular case, the objective is represented by an ORT-H with 19 leaves, each defining a unique feasible polyhedron over $\mathbf{x}$. A truncated version of the tree, with four leaves visible, is shown in Figure 2-10. The $1-\mathrm{R}^2$ error of the approximation is 1.4×10^{-5} over 532 samples.

$$f(\mathbf{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ - 1.5079x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2). \tag{2.20}$$

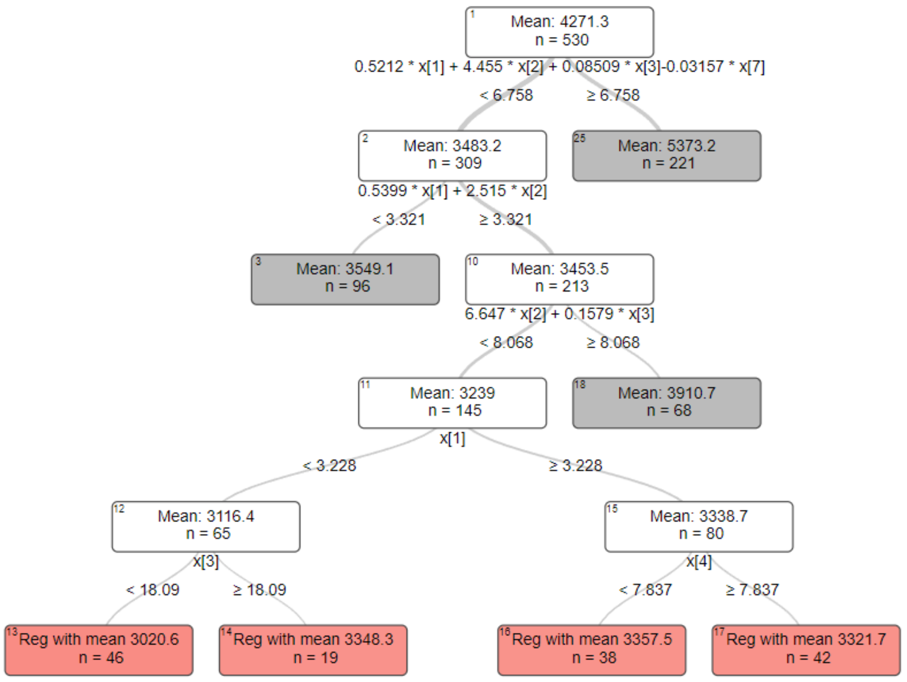


Figure 2-10: The objective function $f(\mathbf{x})$ is approximated via an ORT-H with 19 leaves (4 leaves shown) and $1 - \mathrm{R}^2$ error of 1.4×10^{-5}.

2.7.2 Satellite OOS problem

We test our method on the previously-unsolved optimization problem of satellite on-orbit servicing (OOS) scheduling. Satellite OOS is a future technology that seeks to improve the lifetime of existing and next-generation satellites by allowing autonomous servicer spacecraft to perform repairs or refuels in orbit [61]. OOS is a difficult scheduling problem that acts on a highly nonlinear dynamical system. It is a good problem to address via our method since, in its full MINLP form, the problem is a nonconvex combinatorial optimization problem with nonlinear equality constraints. In addition, due to the 11 orders of magnitude difference in the ranges of decision variables, it is numerically challenging. Before this thesis, it was addressed only via enumeration [61]. Please refer to Appendix A.4 for more details on the full list of constraints; a succinct summary of the problem follows.

The dynamical problem is the orbital mechanics of moving a servicer satellite between client satellites in the same orbital plane. Orbital transfers involve using on-board thrusters to get the servicer into a different orbital altitude than the client satellite, called the phasing orbit, in order to reduce the true anomaly (angular phase difference in radians) between the servicer and the client. The servicer then propels itself back onto the client's orbit to meet the client satellite at the right time and position in space, while obeying conservation of energy, momentum and mass. The scheduling problem involves both choosing the optimal order in which to serve each client satellite (discrete decisions), as well as choosing the optimal phasing orbits (continuous decisions).

In this section, we consider a simple example of OOS. We schedule a single servicer satellite to refuel 7 client satellites in orbit, traveling between clients using on-board propulsors. Each client requires different amounts of fuel, and we constrain the servicer to fulfill its mission in 0.35 years, with the objective being to minimize the wet mass (the dry mass and fuel) of the servicer. The problem parameters are in Table 2.8.

Parameter	Value	Units
Servicer dry mass	500	kg
Propulsor specific impulse	230	(Ns)/kg
Number of client satellites	7	-
Client satellite altitude	780	km
Servicer satellite altitude range	[760,800]	km
Maximum service time	0.35	years

Table 2.8: OOS problem parameters.

The fuel requirements shown in the Figure 2-11 were randomly generated and reflect a possible distribution of fuel needs for client satellites that are part of the same constellation and were launched concurrently at a previous point in time.

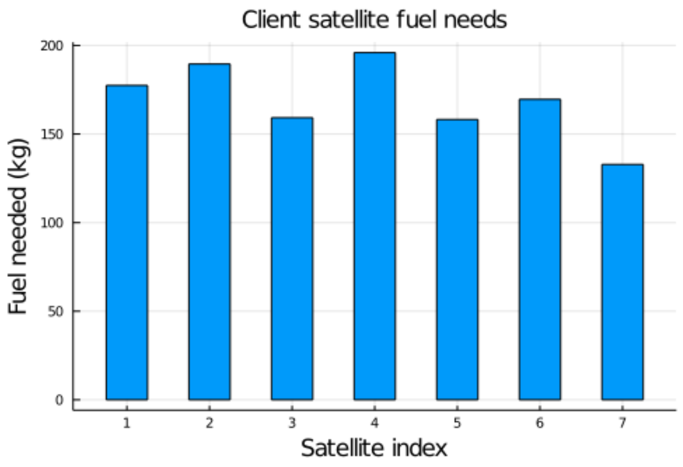


Figure 2-11: Client satellites require different amounts of fuel, which affects the optimal schedule for servicing.

In addressing the OOS problem, we make the following realistic simplifying assumptions, although our method does extend to more general cases. The servicer satellite is delivered by an external rocket to the first client, and uses its own propulsor to use Hohmann transfers between the subsequent client satellites. Thrusting and refueling steps take a negligible amount of time relative to maneuver steps. All client satellites are in the same orbital plane, at the same altitude, and are evenly spaced around the orbit.

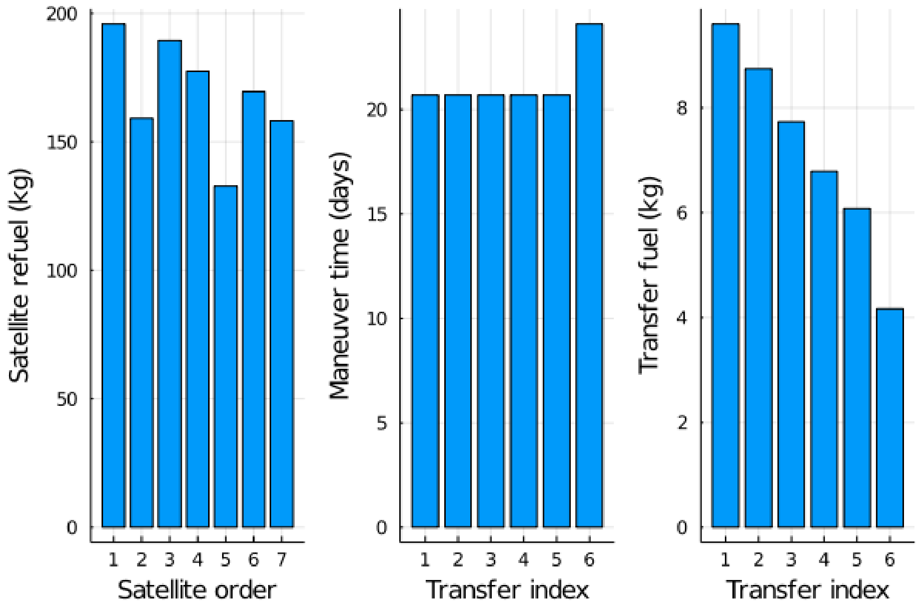
The initial problem of servicing $n_s = 7$ clients has 141 variables, of which $n_s^2 = 49$ binary variables denote the servicing order. The continuous decision variables in nonlinear constraints are bounded from above and below to be compatible with OCT-HaGOn as defined in Section 2.4.1. There are 41 linear constraints in the model representing a subset of the system dynamics. On top of the linear constraints, we have $10(n_s - 1) = 60$ nonlinear constraints, all of them equalities. The constraints are presented in detail in Appendix A.4.

We solve the problem in two ways. First we solve it via OCT-HaGOn. Since we know the constraints of this problem explicitly, we use the ORT-H approximation method as described in Section 2.4.4, separating nonlinearities from affine components of constraints for improved accuracy, and training a tree for each set of recurrent constraints. The resulting MIO problem has 999 continuous and 349 binary variables, and 3650 linear inequalities and 286 linear equalities.

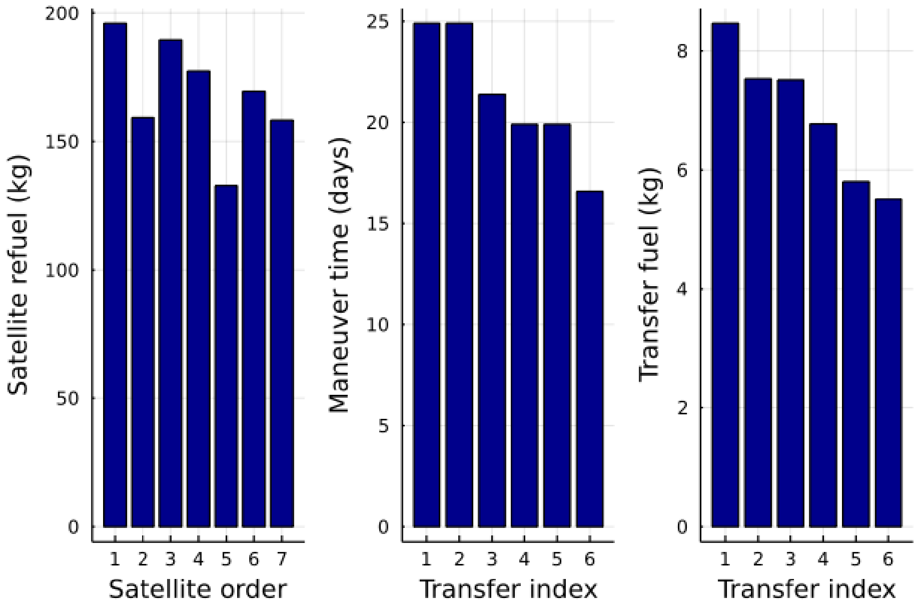
Other global optimizers such as CONOPT, IPOPT and BARON cannot be used as benchmarks for OCT-HaGOn on this particular problem. Since OOS is a mixed-integer problem, gradient-based optimizers such as CONOPT or IPOPT are rendered ineffective, and BARON does not support the nonlinearities present in orbital dynamics. Instead, we successfully discretize out a subset of the nonlinearities in constraints by restricting the possible transfer orbits into 1 km bins. This reduces the complexity of the OOS problem to a MI-bilinear problem, which we are able to solve via Gurobi's MI-bilinear optimizer [42]. The MI-bilinear representation has 394 variables, of which 289 variables are binary. 36 of the 60 nonlinear constraints are turned into bilinear equalities, while the rest are transformed into linear constraints. The solution of the discretized problem is globally optimal, but guaranteed to be worse than the global optimum of the full MINLP formulation, since a discrete set of orbit altitudes is more restrictive than a continuous set. However, the solution is granular enough to be a good benchmark for OCT-HaGOn.

The results are presented in Table 2.9, and shown graphically in Figure 2-12. Firstly, we look for two important effects, demonstrated well by the MI-bilinear solution and easily seen in Figure 2-12. The first is that it is best to refuel satellites with the largest refuel requirements first, since a lighter servicer requires less fuel to transfer between subsequent clients. The second is that it is better to spend more time transferring in the beginning of the mission than the end, since transfers spend less fuel when the servicer is lighter. This is exhibited by a general downward trend in both maneuver times and fuel costs in the MI-bilinear solution.

While OCT-HaGOn properly captures the optimal satellite schedule, it isn't able to find the optimal set of phasing orbits. This is easily seen by observing the flat profile of maneuver times in the OCT-HaGOn solution in Figure 2-12a, which is suboptimal (by $< 0.1\%$ total fuel) to the decreasing profile seen in the discretized solution in Figure 2-12b. In addition, due to the presence of many nonlinear equalities, the PGD method was not able to reduce the infeasibility and optimality gaps, getting stuck in a local optimum. With a maximum tree



(a) The OCT-HaGOn solution.



(b) The MI-bilinear solution.

Figure 2-12: While it captures the orbital dynamics well, OCT-HaGOn is not able to schedule the phasing orbits as well as the MI-bilinear formulation.

Metric	Values						
OCT-HaGOn solution							
Wet mass (kg)	1725.9						
Total maneuver time (years)	0.350						
Satellite order	4	3	2	1	7	6	5
Refuel mass (kg)	196.0	159.2	189.5	177.4	132.9	169.6	158.2
Transfer orbit altitude (km)		765.8	765.8	765.8	765.8	765.8	767.6
Maneuver fuel (kg)		9.60	8.74	7.73	6.79	6.08	4.17
Maneuver time (days)		20.7	20.7	20.7	20.7	20.7	24.1
Orbital revolutions		297.0	297.0	297.0	297.0	297.0	345.3
Discretized MI-bilinear solution							
Wet mass (kg)	1724.4						
Total maneuver time (years)	0.350						
Satellite order	4	3	2	1	7	6	5
Refuel mass (kg)	196.0	159.2	189.5	177.4	132.9	169.6	158.2
Transfer orbit altitude (km)		768.0	768.0	766.0	765.0	765.0	762.0
Maneuver fuel (kg)		8.46	7.53	7.51	6.77	5.80	5.51
Maneuver time (days)		24.9	24.9	21.4	19.9	19.9	16.6
Orbital revolutions		357.1	357.1	306.1	285.7	285.7	238.1

Table 2.9: The discretized and OCT-HaGOn formulations come up with the same optimal satellite schedule, although the discretized solution performs 0.1% better.

depth of 6, the solution has a maximum relative error of 3.5×10^{-3} and a mean relative error of 2.5×10^{-4} on all nonlinear constraints. While this is sufficiently accurate for conceptual design purposes, greater accuracy and a more robust repair procedure are desired.

In terms of solution time, OCT-HaGOn took 14.2 seconds when solved using a personal computer with an 8-core Intel i7 processor. That includes all sampling, evaluation, training and optimization steps. In comparison, the MI-bilinear solution took 17.7 seconds, just for the optimization step. This is in addition to the two days spent by an experienced engineer, reformulating the problem to be compatible with existing efficient optimization formulations.

Despite the suboptimal solution of OCT-HaGOn to the OOS problem, we argue that it is a strong demonstration of the capabilities and promise of the method, especially considering the problem complexity. Notably, OCT-HaGOn successfully finds the optimal satellite servicing schedule, which is arguably the most important decision in the problem. This is despite the fact that the problem is ill-conditioned, with 11 orders of magnitude difference in decision variable values, and has 60 nonlinear equality constraints coupling a majority of the decision variables. In addition, discretized reformulations of such complex global optimization problems may not exist in general. Even if they do, they may be intractable due to the combinatorial nature of such reformulations. To the best of our knowledge, this makes OCT-HaGOn the only global optimization tool in the literature that can address this problem directly.

2.8 Discussion

In this section, we discuss the results and limitations, and propose areas for future work.

2.8.1 Limitations

The proposed method shows promise in solving a variety of global optimization problems, but it is a work in progress. Here we detail some of the limitations of the method as implemented in this thesis; we list these in order from the most to the least significant, in the author's opinion.

While the OOS example demonstrates that the method can address problems with a high degree of nonlinear coupling between decision variables, individual nonlinear constraints involving a large number of active variables will pose challenges in both the OCT-H training time, as well as the accuracy of the tree approximations. Tree accuracy directly affects the quality of the approximate optima. Separability, as described in Section 2.4.4, can partially mitigate this problem, by allowing many nonlinear constraints to be decomposed into linear components and better approximated via a series of ORT-Hs.

In addition, we have yet to rigorously test how solution time and quality scale with the number of variables and nonlinear constraints, and the sparsity of the nonlinear constraints. Given that the performance of OCT-HaGOn is formulation-dependent, there is much to be gained, both in terms of solution time and quality, through formulations that premeditate where OCT-H approximations need to be used, and use them judiciously. We expect OCT-HaGOn to be particularly effective when a majority of the constraints in the optimization problem are linear or convex and therefore efficient, and the constraint learning approach is implemented on the otherwise intractable constraints, with reasonably tight decision variable bounds.

As noted in Section 2.6, the proposed method has no guarantees of global optimality since it is approximate. Thus, different iterations of the method generate high-performing solutions that are locally optimal, but do not have guarantees of global optimality such as those provided by BARON. In addition, while the method is agnostic about whether constraints are explicit or inexplicit, the method has so far been tested on explicit constraints only. This is because of the inavailability of numerical benchmarks with black box functions, due to their incompatibility with other existing global optimizers.

An implicit assumption of the method is that the intractable constraints are quick to evaluate; if this assumption is not true, then the implementation may need to change to accommodate computational requirements. Additionally, the PGD method requires that the constraint functions are auto-differentiable. While this is a modest assumption, it is possible that constraint evaluations do not allow for AD. This could be overcome by finding gradients approximately, e.g. via finite differencing, but this is not currently implemented.

With these limitations outlined, we continue by proposing future work to improve the method.

2.8.2 Decision tree training

The majority of the solution time of OCT-HaGOn is taken by the tree training step.

While the computational cost of training is linear with the number of constraints, the results on benchmarks in Section 2.6 show that training time can vary dramatically depending on the complexity of the underlying constraints. In this section, we discuss several ways to manage computational time.

The first potential source of training time reduction comes from tuning the base tree parameters described in Table 2.1 and implemented in IAI. To do so, we can reduce the complexity of the trees, by reducing the maximum depth and increasing the minbucket parameters. Otherwise we can modify the number of random restarts in tree training. Since the local search method used in generating OCT-Hs and ORT-Hs is locally optimal, we can reduce training time by changing the number of random restarts of candidate trees, as well as the number of random hyperplane restarts. However, both methods have a clear negative tradeoff with respect to the accuracy of the OCT-H approximations. In general, we find that using 10 random tree restarts and 5 hyperplane restarts, as described by the base tree parameters in Table 2.1, we are able to generate trees that are sufficiently accurate for decision making while being efficient enough to use in a real-time optimization setting.

A potentially large source of training time reduction is from recognizing the common form of constraints in a problem. If a nonlinear constraint $g(\mathbf{u}_i) \geq 0$ is repeated k times with different variables $\mathbf{u}_i \subset \mathbf{x},\ i \in [k]$, the constraints can be approximated jointly. Specifically, we can train a single OCT-H to approximate the constraint over the domain $\cup_{i=1}^{k}\text{dom}(\mathbf{u}_\text{i})$. We then express the k constraints as k repetitions of the disjunctive representation of the tree with different variables $\mathbf{u}_i$. In this paper, many benchmarks in Section 2.6 exhibit this kind of repeating behavior, but we treat the constraints as black boxes and do not take advantage of potential speed-ups. For the OOS problem however, we use our knowledge of the constraints to train the trees jointly.

There is also potential in exploiting the noise-free nature of data over explicit constraints to speed up the training process. Currently, training time scales exponentially in the number of features of the data (i.e. number of variables in each constraint), making tree approximations of constraints dense in $\mathbf{x}$ slow. One could speed up the training process by trying a greedy approach, building trees with hyperplanes in a locally optimal manner similar to CART [17], instead of a globally optimal manner via local search heuristics [11]. Another approach could devise specific local search heuristics that sample constraints and train trees in a dynamic manner, in order to speed up training and also reduce the approximation error.

There are also improvements that could be made considering computing architecture. Since individual constraints are learned separately, the training process could be done in parallel, making the best of use of available computational resources. The trees can be efficiently stored once trained, allowing the same trees to be used in different instances of the same optimization problem. This avoids the need to retrain trees, and also avoids having to store the samples required to train the trees, saving on memory. We have developed such methods for development purposes.

2.8.3 Complexity of the MIO approximation

As aforementioned, the complexity of solving the MIO approximations of global optimization problems is modest, since the scale of the MIO is small compared to the abilities of commercial solvers such as Gurobi or CPLEX. However, it is important to note how the complexity of the MIO can scale depending on the number of nonlinear constraints and the depth of the approximating trees.

We first consider the number of auxiliary variables required to pose the MIO approximation. The number of variables used to approximate a nonlinear constraint is a linear function of the number of disjunctive polyhedra describing the feasible space of $\mathbf{x}$, as well as the number of decision variables in the constraint. More explicitly, the total number of binary variables required to approximate the problem is linear with respect to the number of leaves in the decision trees, and equivalent to

$$|L_f| + \sum_{i \in I} |L_{i,1}| + \sum_{j \in J} |L_j|,$$

where L_f, $L_{i,1}$ and L_j are the set of feasible leaves in the objective-, inequality- and equality-approximating trees respectively. In addition, we introduce a number of continuous auxiliary variables. The number of auxiliary variables is equivalent to:

$$1 + |L_f|(p_f + 1) + \sum_{i \in I} \Big(|L_{i,1}|p_i\Big) + \sum_{j \in J} \Big(|L_j|p_j\Big),$$

where p_i is the number of variables in the ith constraint. The maximum number of leaves of a tree is 2^d, so in the worst case, the number of auxiliary binary variables in the problem is $\mathcal{O}(2^d(1 + |I| + |J|))$, and the number of auxiliary continuous variables is $\mathcal{O}(2^d(1 + |I| + |J|)\dim(\mathbf{x}))$, equivalent to the number of binary variables augmented by the dimension of $\mathbf{x}$. In practice however, this worst case is not seen, as the trees are pruned during the training process, and approximated intractable constraints are sparse in $\mathbf{x}$.

The number of disjunctive constraints is more complicated, since the trees are not guaranteed to be of uniform depth, and we do not know a priori the fraction of feasible leaves for a classification tree. However, if we assume that each tree has a depth d_i, we get the following worst case number of disjunctive constraints, not including the univariate bounding constraints for the continuous auxiliary variables:

$$\Big(2^{d_f} \times (d_f + 1)\Big) + 3 + \sum_{i \in I} \Big((2^{d_i} - 1) \times d_i\Big) + \sum_{j \in J} \Big((2^{d_j} - 1) \times d_j\Big) + 2|I| + 4|J|.$$

The above implies that the number of disjunctive constraints in the MIO is $\mathcal{O}(2^d d(1 + |I| + |J|))$, where d is the maximum depth of all approximating trees. This shows the super-exponential impact of tree depth on MIO complexity, where the need for greater accuracy may result in large computational cost. However, for the small to medium scale instances we have considered in this chapter, this is an acceptable tradeoff.

Additionally, the number of variables grows linearly with number of constraints, which could result in the solution time of OCT-HaGOn being exponential in the worst-case. Unlike linear or convex optimization problems, where the average solution time can be sublinear with the number of constraints, OCT-HaGOn is expected to have on average super-linear solution time with respect to number of constraints due to the combinatorial nature of the approximations. We have yet to observe problems that exhibit such exponential-time behavior, likely because of the sparsity of the approximating constraints, and also due to the locally-idealness of the formulation. However, tree complexity needs to be investigated as OCT-H approximations are applied to large scale problems.

2.8.4 Extending to MI-convex formulations

The OCT-HaGOn approach allows us to generate efficient MIO representations of nonlinear constraints that are not efficiently optimizable, i.e. not linear or convex. It opens up the possibility to include these approximations in more general MI-convex formulations, where the efficient convex nonlinear constraints are preserved, either via direct insertion or via outer approximation, while the intractable constraints are approximated via OCT-Hs. This will significantly improve both the speed and accuracy of our method.

2.8.5 Comparing the big-M free and big-M disjunctive formulations

While OCT-HaGOn implements the locally ideal, big-M free disjunctive representations of decision trees as described in Section 2.4.4, it is possible that a big-M representation is faster to solve via commercial solvers, due to the large number of auxiliary variables added in the big-M free approach. It remains to be tested whether it is more efficient to solve the locally ideal but much larger MILO resulting from the big-M free approach, or whether it is more efficient to solve the smaller but non-ideal MILO resulting from the big-M approach. While we have no definitive proof of the relative performance of the two approaches, the author's intuition would point towards a tradeoff based on problem size; it is likely that a big-M approach will outperform the big-M free approach when addressing larger global optimization problems with more nonlinear constraints.

2.8.6 Improved random restarts

As aforementioned, since the constraint learning approach is approximate, random restarts may be required gain confidence in the quality of the locally optimal solutions. Currently, random restarts for OCT-HaGOn involve retraining trees over all nonlinear constraints, and replacing them simultaneously. A better method would be to train an ensemble of trees on each constraint, and permute the tree approximations to generate a set of MI approximations of the problem. The solution of each permutation would provide a near-optimal seed for a new PGD sequence. This would reduce the computational burden of random restarts and result in higher-performing populations of solutions, giving increased confidence in the method.

2.8.7 Optimization over data-driven constraints

There are global optimization contexts where constraints are informed by data, without having access to the underlying models. Some examples are simulation data in the design of engineered systems, outcomes of past experiments, or anthropogenic data such as clinical data and consumer preferences. In theory, OCT-HaGOn is able to learn constraints from arbitrary data and integrate these models in an optimization setting. However, we have yet to perform experiments to confirm the efficacy of OCT-HaGOn in real-world decision making using data-driven constraints. Such an embedding of data into optimization via constraint learning has important implications for a variety of fields, such as healthcare and operations research.

2.8.8 Integration of other MIO-compatible ML models

While this thesis focuses on the use of OCT-Hs and ORT-Hs for constraint learning, there are other ML models that have optimization-compatible representations. Maragno et al. [65] explore the possibility of using linear models, decision trees and their variants, and multi-layer perceptrons to learn constraints and objectives from data. OCT-HaGOn could easily be extended to accommodate such other MIO-representable ML models.

2.9 Conclusion

In this chapter, we have proposed an intuitive new method for solving global optimization problems leveraging interpretable ML and efficient MIO. Our method approximates explicit and inexplicit nonlinear constraints in global optimization problems using OCT-Hs and ORT-Hs, using the natural disjunctive representation of decision trees. We demonstrate, both theoretically and practically, that the disjunctive MIO approximations are efficiently solvable using modern solvers, and result in near-optimal and near-feasible solutions to global optimization problems. We then improve our solutions using gradient-based methods to obtain feasible and high-performing solutions. We demonstrate that our global optimizer OCT-HaGOn is competitive with other state-of-the-art methods in solving a number of benchmark and real-world problems. The Julia implementation of OCT-HaGOn as described in this thesis is available via the link in Appendix A.1.

The method we present is more than a new tool in the global optimization literature. Tree-based optimization stands out among existing global optimization tools because it can handle constraints that are explicit and inexplicit, and even learn constraints from arbitrary data. To the author's best knowledge, it is the most general global optimization method in the literature, since it has no requirements on the mathematical primitives of constraints or variables. Our method only requires a bounded decision variable domain over the nonlinear constraints. This has important implications to a number of fields that can benefit from optimization, but have yet to do so due to lack of efficient mathematical formulations. Some of the relevant aerospace applications are discussed in Section 4.2.

Chapter 3

Optimal Engineering Design Decisions Under Uncertainty

Aerospace design exists in a niche of design problems where “failure is not an option"[1]. This is remarkable since aerospace concepts are rife with uncertainty about technological capabilities, environmental factors, manufacturing quality, and the future state of markets and regulatory agencies. In addition, given certain technologies and configurations, aerospace concepts continue to approach the limits of the Second Law of Thermodynamics, challenging designers to squeeze ever-diminishing performance out of each unit of energy. These two factors broadly explain the high risk of aerospace programs, and the dramatic growth in product development time observed in Section 1.1.

Optimization under uncertainty seeks to provide designs that are robust to realizations of uncertainty in the real world. It has been identified as an area of opportunity for aerospace design in multiple review papers [95, 93], and can play a pivotal role in reducing the high risk of aerospace programs by explicitly considering uncertainty in the design process. We detail some of its potential benefits below.

The aerospace industry has relied heavily on legacy design methods and prior experience when faced with risky design propositions. Legacy tools have been predominant even in the design of novel configurations where experience in and understanding of the design tradespaces is lacking. Since new tools for design under uncertainty will better evaluate risk than legacy tools, there will be increased confidence in and uptake of new design tools.

Design under uncertainty will allow for a better understanding of the tradeoff between risk and performance. Optimization tools that rigorously consider uncertainty will yield designs that are less conservative than traditional designs while meeting the same reliability requirements. These tools will also better evaluate the viability of new concepts and configurations relative to legacy methods, since they will capture the effects of technological uncertainty.

Finally, design under uncertainty will enable guarantees of constraint satisfaction under uncertainty. Designs will be protected against uncertainties in manufacturing quality, environmental factors, technology level, markets, and even requirements. Robust designs can thus avoid costly redesigns, which may be required as parameters in these domains evolve

[1]Quoting Gene Kranz, the mission director of Apollo 13.

during the course of an aerospace program.

In economics, the idea that risk is related to profit is well understood and leveraged. In aerospace engineering however, we often forget that risk aversity necessarily results in lower performance. Considering that conceptual design hedges against program risk, the tractable RO frameworks proposed in this thesis will give aerospace engineers the ability to rigorously trade-off robustness to uncertainty with the performance penalties that result.

3.1 Approaches to design optimization under uncertainty

Faced with the challenge of finding designs that can handle uncertainty, the aerospace field has developed a number of methods to design under uncertainty. Oftentimes, aerospace engineers will implement *margins* in the design process to account for uncertainties in parameters that a design's feasibility may be sensitive to, such as material properties or maximum lift coefficient. Another traditional method of adding robustness is through *multi-mission design* [94], which ensures that the design is able to handle multiple kinds of missions in the presence of no uncertainty. This is a type of *finitely adaptive* optimization geared to ensure performance in off-nominal operations.

These legacy methods have several weaknesses. They provide no quantitative measures of robustness or reliability [95]. They rely on the expertise of an experienced engineer to guide the design process, without explicit knowledge of the tradeoff between robustness and optimality [93]. This is a dangerous proposition especially in the conceptual design phase of new configurations, since prior information and expertise are not available. In these scenarios, it is especially important to implement physics-based tools to explore the design space [94]. Furthermore, legacy methods are often too conservative, ruling out potentially beneficial technologies and configurations due to the inability to adequately trade off performance and risk.

There are two rigorous approaches to solving design optimization problems under uncertainty, which are stochastic optimization (SO) and RO, contrasted in Figure 3-1 and defined below. Note that stochastic optimization is an overloaded term, and exists in at least two contexts in the literature. The first is the solution of deterministic problems with stochastic search space exploration. The second is the solution of design optimization problems with stochastic parameters, which is the focus of this chapter.

SO pairs well with gradient-based approaches to solving nonlinear optimization problems such as those defined in [37], [56] and [57]. These approaches implement an iterative process where the objective function and constraints are evaluated over an initial design, and first- and/or second-order information are used to converge the design towards a local optimum. In this context, SO problems deal with uncertainty by including probability distributions of the uncertain parameters in the iteration, and propagating the distributions through the physics of a design problem to ensure constraint feasibility with certain probability. The predominant goal of SO is to optimize some distributional characteristics, e.g. the mean as in Figure 3-1, of the probability density function of the objective [26].

There have been recent developments in multi-mission aircraft design using SO. Liem et al. [56] propose the use of optimally weighted objective functions over an aircraft's operational design envelope for robust aircraft design. In following work, Liem et al. generate

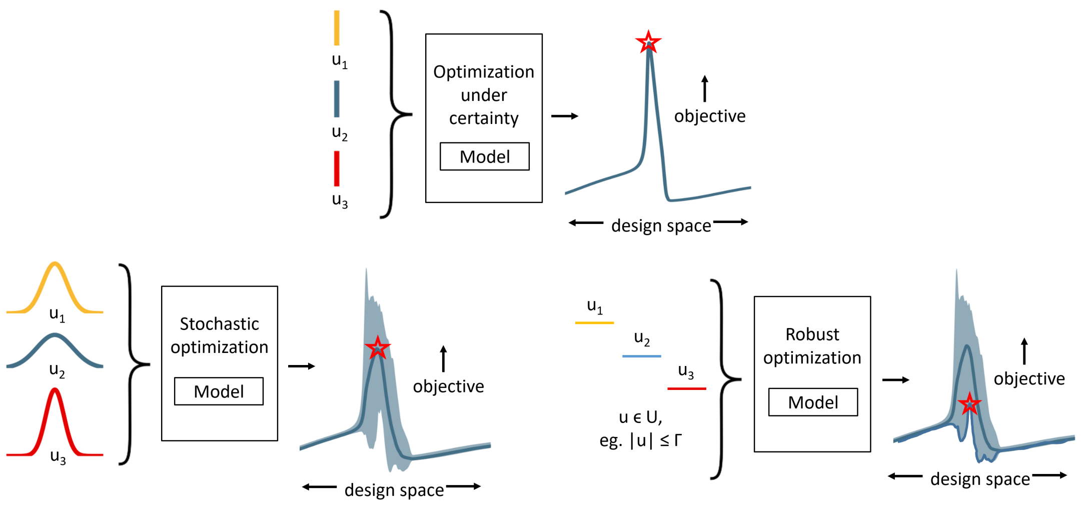


Figure 3-1: SO and RO are methods for optimization under uncertainty that use different definitions of uncertain inputs and produce different objective outcomes.

probability distributions of uncertain parameters from data and minimize the expectation of an objective function over parameter distributions [57]. Although these stochastic methods demonstrate significant improvements over legacy design methods in terms of design robustness, they do not address many of the aforementioned challenges of legacy design methods in capturing the robustness-optimality tradeoff. The scope of the design problems is narrow and limited to aerostructural optimization, and the number of uncertain parameters is low. The formulations assume the presence of data, limiting the effectiveness of the methods in conceptual design. They have large computational costs that are somewhat mitigated through surrogate modeling, but would be detrimental in the conceptual design phase. Most importantly, they lack rigorous mathematical assessments of design feasibility under uncertain parameters.

In contrast to SO, RO can only be applied to mathematical programs that have a robust counterpart, such as linear, quadratic, semidefinite and geometric programs. RO takes a different approach than SO in both the form of uncertain inputs and the objective functions. RO produces designs that are immune to constraint violations as long as parameter values come from within a defined uncertainty set. The objective of RO is to optimize the worst-case objective outcome of a design for a given set over the uncertain parameters. As such, RO avoids the need to sample and propagate probability distributions, and turns SO problems into deterministic problems that are efficiently solved.

3.1.1 Comparison of robust and stochastic optimization for design

Both RO and SO have relative advantages in implementation. Here we reiterate arguments in [74] that the formulation of conceptual engineering design problems under uncertainty as RO problems has advantages over SO formulations (a more mathematical programming centric comparison is made in [8]).

Generality and tractability

In the context of engineering, we claim that an optimization method is general when it can be used to solve a range of problems of interest. On the other hand, tractability describes whether or not the problems are solved to a satisfactory optimum within reasonable computational time. Optimization under uncertainty is a difficult task that puts these two desirable subjective traits at odds with each other.

SO has the advantage of generality. SO methods are easily applicable to black box models or input-output systems. They require little knowledge, if any, about the constraints in the system of interest. RO methods are less general, since they require the design objective and constraints to be explicit and cast in a form that has a worst-case counterpart. Thus models for RO have to be transparent, and RO cannot be applied to black box models without significant prior data manipulation and fitting at a minimum. A mitigating factor is that many classes of conceptual engineering design problems can be cast or approximated in a form that is compatible with robust optimization.

On the other hand, RO is more tractable than SO due to the difference in method of uncertainty propagation. As mentioned previously, SO methods involve the propagation of probability densities throughout a model to determine their effects on constraint feasibility and the objective function. This requires the integration of the product of probability distributions with potential outcomes, and since the integration of continuous functions is difficult, this is often achieved through a combination of high-dimensional quadrature and discretizations of the uncertainty into possible scenarios. This propagation method results in a combinatorial explosion of possible outcomes which need to be evaluated to determine constraint satisfaction and the distribution of the objective. As a result, few problems can be addressed purely through SO (e.g., recourse problems [49, 43]; the energy planning problem [76]; and certain aircraft design problems [56, 57]), and even these are limited by combinatorics and costly system evaluations. Furthermore, they require problem-specific approximations, so that generality is compromised. Robust versions of tractable optimization problems are not guaranteed to be tractable, but in practice the aforementioned classes of optimization problems have tractable robust formulations [8]. In RO, there are no separate optimization and evaluation loops by construction, and thus RO problems can be solved to optimality many orders of magnitude faster than SO problems of the same form [8].

Conceptual design optimization values both generality and tractability, the former because engineers would like to apply methods for optimization under uncertainty without significant mathematical groundwork, and the latter because fast solution times are critical to reduce program risk early on in the design process when more aspects of the design are fluid. From this perspective, the relative intractability of SO-based approaches makes them unreliable for conceptual design, since significant time is needed both to develop problem-specific tractable formulations, and to find satisfactory optima. Furthermore, many engineering design problems such as aircraft design are approximable by optimization forms that have tractable robust counterparts, making RO better suited to conceptual design.

Use of data

SO problems generally require complete knowledge of the probability distributions of parameters. RO requires only 'modest assumptions about distributions, such as a known mean and bounded support' [23]. Since RO does not require as much information about uncertain parameters as SO does, it can better address conceptual design problems where there is a lack of experience, or sparse and noisy data [8]. It is arguable that RO leaves a lot on the table by not taking advantage of distributional information, however there is a growing body of research on distributionally robust optimization [12] which seeks to leverage existing data.

Stochasticity and probabilistic guarantees

Although RO problems solve problems with uncertainty, RO formulations result in *deterministic*[2] solutions that are immune to all possible realizations of parameters in an uncertainty set [8]. There is extensive literature on RO methods that offer differing levels of conservativeness [13] depending on the kind of uncertainty set considered, that are guaranteed to be feasible over the uncertainty set of interest.

SO formulations provide no probabilistic guarantees since the optimum depends on realizations of random variables [84]. This is not satisfactory from an engineering perspective, since optimization runs over the same parameters may result in different solutions. Furthermore, designs can be sensitive to issues in sampling schemes over potentially unknown probability distributions. In the context of engineering design, the determinism and probabilistic guarantees of RO make it superior to SO.

It is important to highlight that, although both RO and SO seek to address the problem of optimization under uncertainty, they solve fundamentally different problems. In an ideal world where we have a problem that is tractable and globally optimal for both methods, the two different approaches would result in different solutions.

3.1.2 Geometric and signomial programming for engineering design under uncertainty

Formulations exist for solving the robust geometric program (RGP) with parametric uncertainty [80]. The creation of a robust signomial program (RSP) framework to capture uncertainty in engineering design, and specifically aircraft design, will allow us to have more confidence in the results of the conceptual design phase, reduce program risk, and increase overall system performance.

3.2 Contributions

In this chapter, we make the following contributions to the literature:

1. We propose addressing engineering design problems with parametric uncertainty using RSPs.

[2]Determinism in this case refers to the outcomes of free variables in the optimization model. Different instances of a deterministic design problem with the same parameters will result in the same solution.

2. Based on the framework developed by Saab and Öztürk [80, 74], we extend the RGP framework to SPs, which we solve as a sequence of RGPs.

3. We demonstrate and quantify the benefits of RO in ensuring design feasibility and performance using Monte Carlo (MC) simulations of the uncertain parameters. We contrast these with the feasibility and performance of the nominal and margin-driven designs.

4. We explore the potential of RO in the multiobjective design setting.

5. We propose a new goal programming method for multiobjective optimization under uncertainty using RO, that uses uncertainty set size as its figure of merit.

We now more specifically define the mathematical principles of robust optimization, before detailing our RSP method.

3.3 Mathematical theory of robustness

Given an optimization problem under parametric uncertainty, we define the set of possible realizations of uncertain vector of parameters $\mathbf{u}$ in the uncertainty set $\mathcal{U}$. This allows us to define the problem under uncertainty below, with objective f_0 and constraints $f_i,\ i \in [n]$ over design variables $\mathbf{x}$ and uncertain parameters $\mathbf{u}$.

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & f_i(\mathbf{x}, \mathbf{u}) \leq 0,\ \forall \mathbf{u} \in \mathcal{U},\ i \in [n]. \end{aligned}$$

In the trivial case when $\mathcal{U}$ has a single element, we recover the deterministic problem where parameters $\mathbf{u}$ are fixed and certain. The problem of interest however has parametric uncertainty over continuous variables, for which $\mathcal{U}$ is a non-empty set with countably infinite members. This general problem is infinite-dimensional, since it is possible to formulate an infinite number of constraints with the countably infinite number of possible realizations of $\mathbf{u} \in \mathcal{U}$.

To circumvent this issue, we can define the following robust formulation of the uncertain problem below.

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \max_{\mathbf{u} \in \mathcal{U}} f_i(\mathbf{x}, \mathbf{u}) \leq 0,\ i \in [n]. \end{aligned}$$

This formulation hedges against the worst-case realization of the uncertainty in the defined uncertainty set. The set is often described by a norm, which contains possible uncertain outcomes from distributions with bounded support:

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \max_{\|\mathbf{u}\| \leq \Gamma} f_i(\mathbf{x}, \mathbf{u}) \leq 0,\ i \in [n], \end{aligned} \tag{3.1}$$

where Γ is defined by the user as a global uncertainty bound. The larger the Γ, the greater the size of the uncertainty set that is protected against.

3.4 Robust signomial programming formulation

This section overviews the proposed RSP formulation. As a refresher of Section 3.3, robust signomial programming assumes that parameter uncertainties are defined by an uncertainty set, and solves a reformulated design problem to find the best solution, through the process shown in Figure 3-2. As long as the original optimization problem is SP-compatible, a robust formulation of the problem exists, making this method general to aerospace problems such as those in [94, 54, 52]. We derive the intractable formulation of a RSP below.

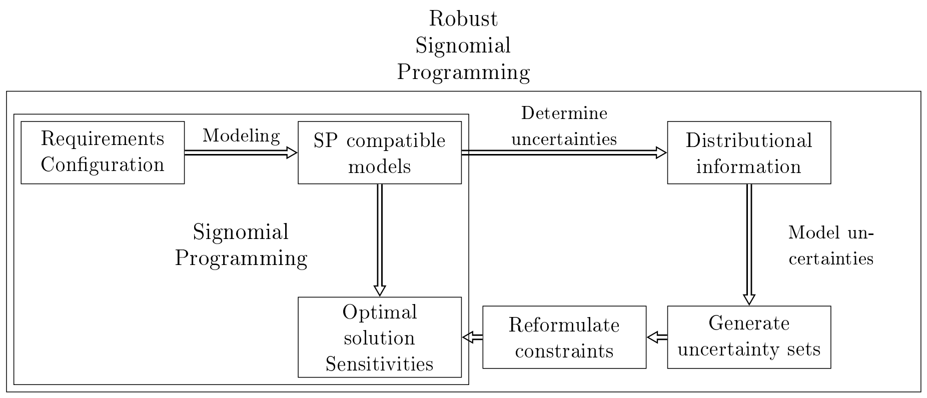

Robust Signomial Programming
Requirements Configuration
Modeling
SP compatible models
Determine uncertainties
Distributional information
Signomial Programming
Model uncertainties
Optimal solution Sensitivities
Reformulate constraints
Generate uncertainty sets

Figure 3-2: A block diagram showing the difference between the design process using a SP and a RSP.

To refresh the reader, *SP in exponential form* is as follows:

$$\begin{array}{ll} \min & f_0(\mathbf{x}) \\ \text{s.t.} & \sum_{k=1}^{K_i} e^{\mathbf{a_{ik}}^\top \mathbf{x} + b_{ik}} - \sum_{k=1}^{G_i} e^{\mathbf{c_{ik}}^\top \mathbf{x} + d_{ik}} \leq 0,\ \forall i \in [m], \end{array} \tag{3.2}$$

where the constraints are represented as difference-of-posynomials in exponential form. Let $\mathbf{a_{ik}}$ and $\mathbf{c_{ik}}$ be the $((i-1)\times m + k)^{th}$ rows of the exponents matrices $\mathbf{A}$ and $\mathbf{C}$ respectively, and b_{ik} and d_{ik} be the $((i-1)\times m + k)^{th}$ elements of the coefficients vectors $\mathbf{b}$ and $\mathbf{d}$ respectively.

The data ($\mathbf{A}$, $\mathbf{C}$, $\mathbf{b}$, $\mathbf{d}$) is assumed to be uncertain and living in an uncertainty set $\mathcal{U}$, where $\mathcal{U}$ is parametrized affinely by a perturbation vector ζ:

$$\mathcal{U} = \left\{[\mathbf{A};\mathbf{C};\mathbf{b};\mathbf{d}] = \left[\mathbf{A}^0;\mathbf{C}^0;\mathbf{b}^0\ \mathbf{d}^0\right] + \sum_{l=1}^{L} \zeta_l \left[\mathbf{A}^l;\mathbf{C}^l;\mathbf{b}^l;\mathbf{d}^l\right]\right\} \tag{3.3}$$

where $\mathbf{A}^0$, $\mathbf{C}^0$, $\mathbf{b}^0$, and $\mathbf{d}^0$ are the nominal exponents and coefficients, $\left\{\mathbf{A}^l\right\}_{l=1}^L$, $\left\{\mathbf{C}^l\right\}_{l=1}^L$, $\left\{\mathbf{b}^l\right\}_{l=1}^L$, and $\left\{\mathbf{d}^l\right\}_{l=1}^L$ are the basic shifts of the exponents and coefficients, and ζ_l is the l^{th}

component of ζ belonging to a perturbation set $\mathcal{Z} \in \mathbb{R}^L$ such that

$$\mathcal{Z} = \left\{\zeta \in \mathbb{R}^L : \|\zeta\| \leq \Gamma\right\}. \tag{3.4}$$

As aforementioned, our goal is a formulation that is immune to uncertainty in the data. Accordingly, the robust counterpart of the uncertain SP in (3.2) is

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \max_{\zeta \in \mathcal{Z}} \left\{\textstyle\sum_{k=1}^{K_i} e^{\mathbf{a_{ik}}(\zeta)^\top \mathbf{x} + b_{ik}(\zeta)} - \sum_{k=1}^{G_i} e^{\mathbf{c_{ik}}(\zeta)^\top \mathbf{x} + d_{ik}(\zeta)}\right\} \leq 1, \ \forall i \in [m]. \end{aligned} \tag{3.5}$$

The optimization problem in (3.5) is intractable using current solvers, therefore a heuristic approach to solving a RSP approximately as a sequential RGP will be presented in the following sections. As our approach is based on tractable robust geometric programming, a brief review of the subject will follow based on [80].

3.4.1 Review of tractable robust geometric programming

This section presents a brief review of the approximation of a RGP as a tractable optimization problem as discussed in [80], using principles of robust linear optimization reviewed in Appendix B.2. The robust counterpart of an uncertain geometric program is

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \max_{\zeta \in \mathcal{Z}} \left\{\textstyle\sum_{k=1}^{K_i} e^{\mathbf{a_{ik}}(\zeta)^\top \mathbf{x} + b_{ik}(\zeta)}\right\} \leq 1, \ \forall i \in [m], \end{aligned} \tag{3.6}$$

which is Co-NP hard in its natural posynomial form [22]. We review the three approximate formulations of a RGP.

Simple conservative formulation

One way to approach the intractability in (3.6) is to replace the max-of-sum by the sum-of-max, leading to the following formulation.

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^{K_i} \max_{\zeta \in \mathcal{Z}} \left\{e^{\mathbf{a_{ik}}(\zeta)^\top \mathbf{x} + b_{ik}(\zeta)}\right\} \leq 1, \ \forall i \in [m] \end{aligned} \tag{3.7}$$

Maximizing a monomial term is equivalent to maximizing an affine function, therefore (3.7) is tractable.

Equivalent intermediate formulations

These formulations are equivalent to the formulation in (3.6), but with smaller, easier to handle posynomial constraints. By the properties of inequalities, the posynomial P in posynomial inequality $M \geq P$ can be divided into an equivalent set of smaller posynomials based on the dependence between its monomial terms. Figure 3-3 shows how a constraint can be represented as an equivalent set of smaller posynomial constraints.

$$\begin{tikzcd}
P & =M1 \arrow[d] & +M2 \arrow[drr] & +M3 \arrow[dl] & +M4 \arrow[dlll] & +M5 \arrow[dl] & +M6 \arrow[d] \\
 & S1 & & & S2 & & S3
\end{tikzcd}$$

$$max\{P\} \leq 1 \iff \begin{array}{cl} t_1 + t_2 + t3 & \leq 1 \\ max\{S_1\} = max\{M_1 + M_3 + M_4\} & \leq t_1 \\ max\{S_2\} = max\{M_2 + M_5\} & \leq t_2 \\ max\{S_3\} = max\{M_6\} & \leq t_3 \end{array}$$

Figure 3-3: Partitioning of a large posynomial into smaller posynomials requires the addition of auxiliary variables. S_i are posynomials with independent sets of variables.

The posynomial constraints are categorized into three sets: large posynomials, two-term posynomials and monomials, represented by $S1$, $S2$ and $S3$ respectively. Monomials are tractable, and two-term posynomials can be well approximated using piecewise-linear functions [47]; thus, we can use techniques for robust linear optimization to address both monomials and two-term posynomials directly. We implement the following two tractable approximations for large posynomials.

In the *Linearized Perturbations* approximation, which can be used when exponents are known and certain, robust large posynomial constraints are approximated as signomial constraints. The exponential perturbations in each posynomial are linearized using a modified least squares method, and then the posynomial is robustified using techniques from robust linear programming. The resulting set of constraints is SP compatible, therefore, a RGP is approximated as a SP.

The *Best Pairs* approximation allows for uncertain coefficients and exponents. In this case, the large posynomials can't be approximated as SP-compatible constraints, and further simplification is needed. This formulation aims to maximize each pair of monomials in each posynomial, while finding the best combination of monomials that gives the least conservative solution. [80] provides a descent algorithm to find locally optimal combinations of the monomials, and shows how the uncertain GP can be approximated as a GP for polyhedral uncertainty, and a conic optimization problem for ellipsoidal uncertainty with uncertain exponents. For a detailed description of the above formulations, please refer to [80]. An algorithm for solving a RSP based on the above formulations is provided in the next section.

3.4.2 Solution of robust signomial programs

This section presents heuristic algorithms to solve a RSP based on tractable RGP formulations in Section 3.4.1

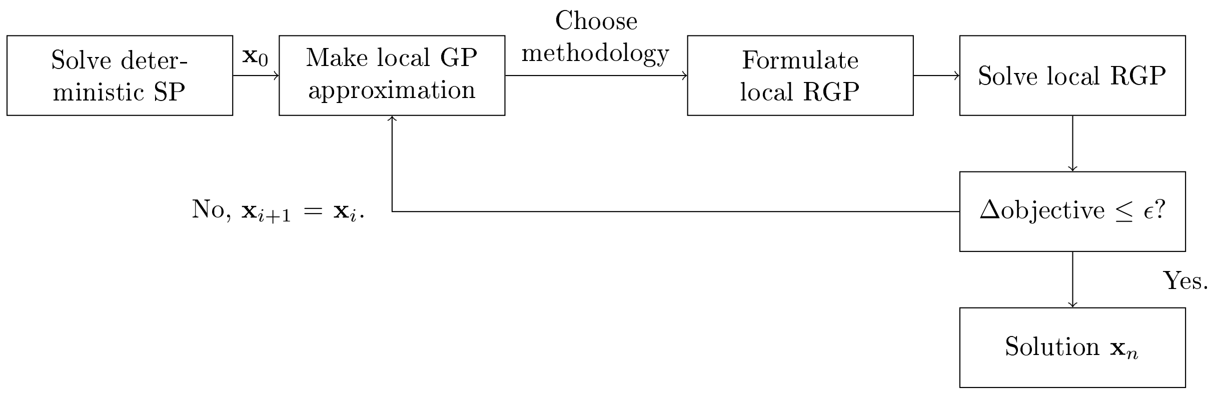


Figure 3-4: A block diagram showing the steps of solving a RSP.

General RSP Solver

A common algorithm to solve a SP involves sequentially solving local GP approximations. Similarly, our approach to solve a RSP is based on solving a sequence of local RGP approximations. In Figure 3-4, we provide a step-by-step algorithm. In this heuristic, a good initial guess will lead to faster convergence and possibly a better solution. The deterministic solution of the uncertain SP is in general a good candidate $\mathbf{x}_0$, but naive guesses also suffice.

For comparisons between methods ahead, we write the algorithm explicitly as follows:

1. Choose an initial guess $\mathbf{x}_0$.
2. Repeat:
   (a) Find the local GP approximation of the SP at $\mathbf{x}_i$.
   (b) Find the RGP formulation of the GP.
   (c) Solve the RGP to obtain $\mathbf{x}_{i+1}$.
   (d) If $\mathbf{x}_{i+1} \approx \mathbf{x}_i$: break.

Any of the methodologies in Section 3.4.1 can be used to formulate the local RGP approximation. However, depending on the RGP formulation chosen to solve the RSP, the formulation and solution blocks in Figure 3-4 are adjusted.

Best Pairs RSP solver

If the Best Pairs methodology is exploited, then the above algorithm changes so that each iteration solves the local RGP approximation and chooses the best permutation for each large posynomial. The modified algorithm is the following:

1. Choose an initial guess $\mathbf{x}_0$.
2. Repeat:

(a) Find the local GP approximation of the SP at $\mathbf{x}_i$.

(b) For each large posynomial constraint, select the new permutation ϕ such that ϕ minimizes the robust large constraint evaluated at $\mathbf{x}_i$.

(c) Solve the approximate tractable counterparts of the local GP in (3.6), and let $\mathbf{x}_{i+1}$ be the solution.

(d) If $\mathbf{x}_{i+1} \approx \mathbf{x}_i$: break.

Linearized Perturbations RSP solver

On the other hand, if the Linearized Perturbations formulation is used, then we can avoid solving a SP at each iteration by first approximating the original SP constraints locally, and in the same loop approximating the robustified possibly-signomial constraints locally, thus solving a GP at each iteration instead of a SP. The algorithm is the following:

1. Choose an initial guess $\mathbf{x}_0$.

2. Repeat:

   (a) Find the local GP approximation of the SP at $\mathbf{x}_i$.

   (b) Robustify the constraints of the local GP approximation using the Linearized Perturbations methodology.

   (c) Find the local GP approximation of the resulting local SP at $\mathbf{x}_i$.

   (d) Solve the local GP approximation in step c to obtain $\mathbf{x}_{i+1}$.

   (e) If $\mathbf{x}_{i+1} \approx \mathbf{x}_i$: break.

3.5 Aerospace problem

We implement the RSP formulation above on an unmanned, gas-powered aircraft design problem that is systematically developed in [72], with the ellipsoidal fuselage model borrowed from [20]. We optimize a wing, fuselage, and engine system given a payload and range requirement. The optimization model captures fundamental tradeoffs in aircraft design, and was developed using GPkit, a Python package that provides abstractions for using GPs in engineering design [19]. The nominal model has 175 variables and 153 constraints, a common level of sparsity for GP and SP models. A short qualitative overview of the model follows; for more detailed information, please refer to [20] and [72]. The uncertainties associated with the parameters will be described in Section 3.6.

3.5.1 Flight profile

The flight profile model is borrowed from [94]. Within the model, the trajectory of the aircraft is optimized over four steady flight segments, although we only model climb segments and therefore the stored gravitational potential energy of the aircraft is not captured.

3.5.2 Atmosphere

The atmosphere model is taken from [88], and considers changes in density and dynamic viscosity with altitude, for a standard atmosphere.

3.5.3 Aircraft

The aircraft is modeled as a wing, fuselage and engine system. The aircraft is assumed to be in steady flight, so that the thrust power is equal to the sum of the drag power and rate of change of potential energy of the aircraft, and the lift is equal to the total weight, ignoring the vertical component of thrust in climb. Its total weight is the sum of its components. The aircraft has to be able to takeoff at specified minimum speed without stalling as well. Aircraft component models are detailed below.

Wing

Lift is generated by the wing as a function of its geometry and free stream conditions. The wing structure model is based on a beam model with a distributed lift load, and a point mass in the center representing the fuselage. Wing fuel volume is modeled as a fraction of the internal volume available in the wing. The weight of the wing is the sum of skin and spar weights. Its drag is the sum of induced and profile drags, the latter of which is constrained by a 3-term softmax-affine posynomial fit [44] of drag polars generated in XFOIL [27]. The airfoil used was designed by Prof. Mark Drela of MIT and is a variant of those implemented in [20].

Fuselage

The fuselage contains the fuel and payload internally, and the engine externally. It is assumed to be ellipsoidal in shape, and its drag is estimated using a form factor. The fuselage is assumed not to contain any structural members, and so its weight consists only of skin weight.

Engine

The aircraft is powered by a naturally aspirated piston engine. It is subject to power lapse at lower air densities at higher altitudes. Engine weight versus maximum sea level power, and brake specific fuel consumption versus thrust and altitude are modeled using the posynomial fits of engine performance data from [73].

Source of non-log-convexity: fuel volume

The signomial constraint in the optimization appears in the aircraft total fuel volume constraint, as shown in (3.8):

$$V_{\mathrm{f}} \leq V_{\mathrm{f_{wing}}} + V_{\mathrm{f_{fuse}}}. \tag{3.8}$$

The signomial constraint makes the problem non-log-convex, which means that the solution methods detailed by Saab [80] need to be extended to accommodate this optimization problem.

Table 3.1: Parameters and uncertainties (increasing order)

Parameters	Description	Value	% Uncert. (3σ)
e	span efficiency	0.92	3
μ	air viscosity (SL)	1.78×10^{-5} kg/(ms)	4
ρ	air density (SL)	1.23 kg/m^3	5
$C_{L,\max}$	stall lift coefficient	1.6	5
k	fuselage form factor	1.17	10
$C_{f,\text{ref}}$	reference fuselage skin friction factor	0.455	10
ρ_p	payload density	1.5 kg/m^3	10
N_ult	ultimate load factor	3.3	15
V_min	takeoff speed	35 m/s	20
W_p	payload weight	3000 N	20
$W_\text{coeff,strc}$	wing structural weight coefficient	2×10^{-5} 1/m	20
$W_\text{coeff,surf}$	wing surface weight coefficient	60 N/m^2	20

3.6 Uncertainties and sets

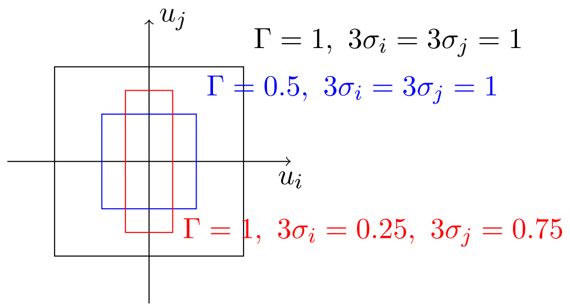
As mentioned in Section 3.1.1, one of the advantages of RO over SO is the fact that it only requires as inputs the uncertainty set size instead of complete probability distributions over each parameter. In the context of this work, the set size is defined relatively in each uncertain design parameter u_i by $3\sigma_i$, and globally by scalar parameter Γ. An illustration of the relationship between 3σ's and Γ is provided in Figure 3-5, and explained in Sections 3.6.1 and 3.6.2.

3.6.1 Design parameter uncertainties

The relative size of the uncertainty set in each uncertain variable is given by three times the coefficient of variation (CV)[3], as listed in Table 3.1. Since for the rest of this work all standard deviations (σ) are normalized by the means of the parameters, we will use 3σ to represent 3CV.

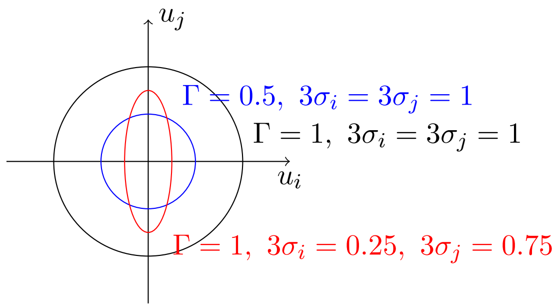
In this case of a conceptual aircraft design with no prior data, the parameter uncertainties reflect aerospace engineering intuition. The wing weight coefficients $W_\text{coeff,strc}$ and $W_\text{coeff,surf}$, and the ultimate load factor N_ult have large 3σ's because the build quality of aircraft components is often difficult to quantify with a large degree of certainty. The payload weight and density (W_p and ρ_p) have large uncertainties since the payload is often developed concurrently with the aircraft. Parameters that engineers take to be physical constants (sea level air viscosity and density, μ and ρ) and those that can be determined with a relatively high degree of accuracy (e) have relatively low deviations. Parameters that require testing to determine ($C_{L,\max}$, $C_{f,\text{ref}}$, V_min) have a level of uncertainty that reflects the expected variance of empirical studies. However, note that these quantities are ultimately picked by the designer using prior experience and data, and the level of conservativeness in the design

[3]The CV is defined as follows: CV $= \frac{\sigma}{|\mu|}$, where σ is the standard deviation and μ is the mean of the parameter.



u_j
$\Gamma = 1,\ 3\sigma_i = 3\sigma_j = 1$
$\Gamma = 0.5,\ 3\sigma_i = 3\sigma_j = 1$
u_i
$\Gamma = 1,\ 3\sigma_i = 0.25,\ 3\sigma_j = 0.75$

(a) Example L∞ or box sets.



u_j
$\Gamma = 0.5,\ 3\sigma_i = 3\sigma_j = 1$
$\Gamma = 1,\ 3\sigma_i = 3\sigma_j = 1$
u_i
$\Gamma = 1,\ 3\sigma_i = 0.25,\ 3\sigma_j = 0.75$

(b) Example L2 or ellipsoidal sets.

Figure 3-5: Γ defines the overall size of norm uncertainty sets, while 3σ defines the relative size of the set in each uncertain parameter.

will be greatly affected by the chosen 3σ's.

3.6.2 Uncertainty sets considered

The robust design problem is solved for box and ellipsoidal uncertainty sets, which are defined by the L∞- and L2-norms, and bounded by varying the parameter Γ. Intuitively, for both sets, Γ is a global measure of how much risk is being hedged against, and affects all parameter uncertainties simultaneously. $\Gamma = 0$ implies that all of the parameters take their nominal values with zero uncertainty, which we call the nominal problem, and larger Γ protects against greater uncertainty. Γ is more rigorously defined in the context of robust linear programming in Appendix B.2.

For box uncertainty, Γ scales the width of the L∞ hypercube as shown in Figure 3-5a, whose dimensionality is the same as the number of uncertain parameters (12). More intuitively, $\Gamma \times 3\sigma_i$ defines the range of the possible values of uncertain parameter u_i, normalized by the mean of u_i. It can be easy to assume that using margins and box uncertainty sets will yield the same designs, but they fundamentally function differently. Firstly, the worst case outcome in box uncertainty can come *from the interior* of the uncertainty set, instead of the corner of the hypercube considered by margins. Furthermore, there is no guarantee (and it is unlikely) that the chosen corner, i.e. particular allocation of margins, is the most conservative point in the uncertainty set. It is even possible that *the wrong sign of margin* is allocated for certain parameters, since SPs are nonlinear and local sensitivities cannot be used reliably to intuit global behavior. Consider in this particular example the sea level air density ρ. Higher air density is better for takeoff performance and naturally aspirated engine performance, but results in higher drag, so it is difficult for a designer to determine how to best allocate margin on ρ. Thus for the rest of this chapter the direction of margins is determined using the local sensitivities of the nominal solution, which are obtained at no extra computational cost in the solution of the terminal GP approximation of the SP. With these considerations in mind, box uncertainty is expected to be strictly more conservative and more appropriate than the use of margins in conceptual design, since (1) margins fail to capture the level of conservativeness they signal, and (2) prior information (in this case the nominal solution) is required to allocate margin effectively.

For ellipsoidal uncertainty, Γ is the maximum diameter of the Euclidian norm ball of $\mathbf{u}$ as shown in Figure 3-5b, where u_i is one-third the number of standard deviations of perturbation of the ith parameter from its nominal value. Ellipsoidal uncertainty exploits the fact that the joint probability of multiple uncertain parameters taking values in the tails of their respective distributions is very low. So while it does not protect deterministically for all outcomes of the uncertain parameters within 3σ, it is expected to protect against uncertain outcomes less conservatively than the box uncertainty set, with little compromise in the ability of the design to satisfy constraints.

3.7 Results

We implement our RSP algorithm on the aforementioned conceptual aircraft design problem. Our implementation is written in Python, and is free and open source; it is available via the link provided in Appendix B.1. Our objective function is total fuel consumption, which is to be minimized given a payload and range requirement.

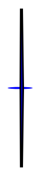
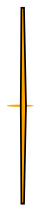
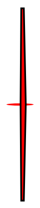
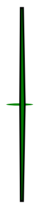
3.7.1 Mitigation of probability of failure

First, the optimization problem is solved in presence of no uncertainty. It is expected that this aircraft has a high probability of failure due to its sensitivity to the outcomes of uncertain parameters. Then, using the sign of sensitivities of the nominal solution, we assign 3σ margins for each parameter and generate a design using margins. These two solutions are compared with RO results for box and ellipsoidal uncertainty sets at $\Gamma = 1$, using the Best Pairs robustification method. From here onward we refer to aircraft designed under no uncertainty, under margins, under box uncertainty and under ellipsoidal uncertainty as 'the nominal aircraft', 'the margin aircraft', 'the box aircraft' and 'the ellipsoidal aircraft' respectively.

The design variables are then fixed for each solution, and the designs are simulated for different realizations of the uncertain parameters. This allows for statistical analysis of design performance, and an estimate of each design's probability of constraint violation, which we define as its probability of failure (PoF). In this MC scheme, the random variables are simulated from independent and identically distributed 3σ-truncated Gaussians. We simulate from the truncated Gaussian since this makes it possible to confirm mathematically that for $\Gamma = 1$, all simulations of 3σ uncertain parameters are deterministically feasible for the box uncertainty set. The results are in Table 3.1. Designs for each solution for the rest of the section are simulated with the same MC samples for consistency.

It is noteworthy in the PoF at the bottom of Table 3.2 that, for the nominal problem ($\Gamma = 0$), only 12 percent of the MC evaluations result in feasible solutions. This means that an aircraft designed for the average case would almost surely fail to satisfy the mission requirements, even with equal likelihood of favorable versus unfavorable uncertain outcomes from the symmetric truncated Gaussian. That being said, depending on the problem, it may necessary to sacrifice performance to achieve a high degree (3σ) of reliability as in the solution for $\Gamma = 1$. Furthermore, the margin aircraft, the box aircraft and the ellipsoidal aircraft spend on average 53%, 55% and 39% more fuel respectively than the aircraft designed

Table 3.2: SP aircraft optimization results, for $\Gamma = 1$.

Free variable	Description	Units	No Uncert.	Margins	Box	Ellipsoidal
L/D	mean lift-to-drag ratio	-	45.0	35.4	36.1	38.4
AR	aspect ratio	-	38.0	25.0	24.6	28.1
Re	Reynolds number	-	8.44×10^5	1.21×10^6	1.35×10^6	1.21×10^6
S	wing planform area	m^2	6.27	14.9	14.6	12.8
τ	airfoil thickness ratio	-	0.175	0.197	0.198	0.192
V	mean flight velocity	m/s	41.7	34.6	35.4	36.2
T_{flight}	time of flight	hr	20.0	24.1	23.6	23.1
W_{w}	wing weight	N	1170	2080	2090	1940
$W_{\mathrm{w,strc}}$	wing structural weight	N	792	1100	1130	1090
$W_{\mathrm{w,surf}}$	wing skin weight	N	376	985	966	851
W_{fuse}	fuselage weight	N	151	192	177	168
W_{e}	engine weight	N	84.4	111	122	115
$V_{\mathrm{f,avail}}$	total fuel volume	m^3	0.0267	0.0458	0.0502	0.0459
$V_{\mathrm{f,fuse}}$	fuselage fuel volume	m^3	0.0134	0	0	0
$V_{\mathrm{f,wing}}$	wing fuel volume	m^3	0.0133	0.0680	0.0667	0.0468
	sketches to scale					

Metric	Description	Units	No Uncert.	Margins	Box	Ellipsoidal
Objective	fuel weight	N	214	367	402	368
E[Objective]	mean fuel weight	N	207	316	320	287
σ[Objective]	std. dev. of fuel weight	N	11	12	12	11
P[failure]	probability of failure	%	88	0	0	0

for the nominal case, but they also are robust to all uncertain outcomes in the 3σ set for the MC simulation.

Table 3.2 also indicates that margins are not a good method of allocating uncertainty. The claim for the use of margins is that they protect against the worst case outcome of each parameter, but the results show otherwise. Since the box design at $\Gamma = 1$ is strictly more conservative (worse worst-case outcome) over the 3σ hypercube than the margin design, we see that a margin from the interior of the hypercube rather than its corner is more effective in protecting against the worst case. Furthermore, there are no probabilistic guarantees that the aircraft with margins would not fail one of the MC simulations. Given enough samples, it is almost surely true that some MC simulations will violate feasibility for the design with margins, whereas RO under box uncertainty guarantees deterministically that the constraints are satisfied.

We also posited that the ellipsoidal aircraft, although it isn't deterministically robust to all 3σ uncertainties, would be less conservative than the margin and box designs while not significantly sacrificing PoF. This is confirmed since the ellipsoidal design fails none of the random samples, and spends 9% and 10% less fuel on average than the margin and box aircraft respectively. The significance of this cannot be understated: the use of ellipsoidal uncertainty results in designs that have strictly better performance outcomes, while protecting against a similar amount of risk as designs using margins or box uncertainty.

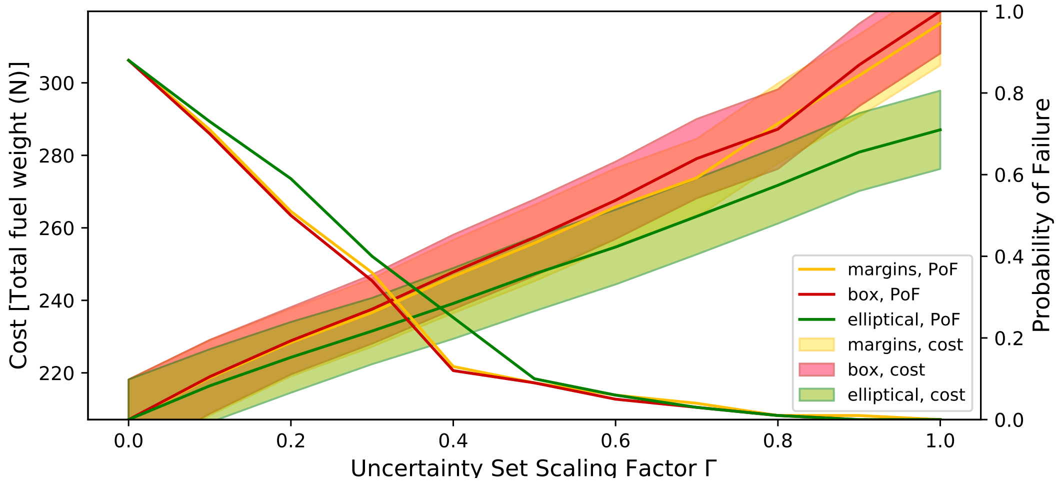


Figure 3-6: Simulated cost and PoF of the optimal margin, box, and ellipsoidal aircraft as a function of Γ. The banded lines represent the mean and standard deviation of total fuel burn, simulated with 100 MC samples of uncertain parameters.

An analysis on the range $\Gamma = [0, 1]$ was performed to confirm that the trends from Table 3.2 hold for all Γ. Figure 3-6 shows that PoF goes monotonically towards zero as Γ increases for all three methods, where box uncertainty is more conservative than ellipsoidal uncertainty over the whole Γ domain, with no such guarantees for margins.

In absolute terms, the nominal SP under zero uncertainty or with margins takes just under 0.9 seconds to solve on a modern laptop computer using Mosek [2], an interior point solver that is free for academic use; we refer the reader to [53] and [94] for more in-depth SP solution time analyses. In Figure 3-7 we examine briefly in relative terms how the different RSP methodologies compare in terms of setup and run times. Since the setup time of the nominal problem is minimal, we have normalized the results by the solution time of the nominal problem. The bottom axis ranks the methods by their level of conservativeness, Best Pairs and Simple Conservative formulations being the least and most conservative respectively, and where the ellipsoidal formulations are less conservative than the box formulations. For this aircraft design problem, the preferred Best Pairs methodology with an ellipsoidal uncertainty set is competitive in solution and setup times relative to other methods, while providing the least conservative solutions. Note that setup and solution times for RSPs are highly problem-specific, so it is not possible to predict the time performance of other RSP-compatible problems from these results. Time performance will vary depending

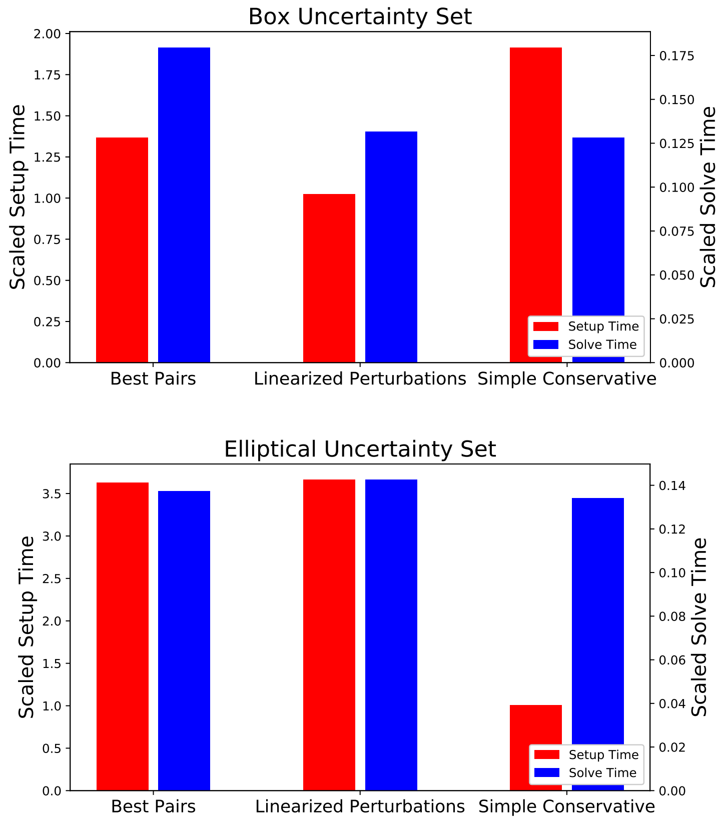


Figure 3-7: Robust aircraft optimization setup and solution times for different RSP approximations, normalized by the nominal problem solution time, for $\Gamma = 1$.

on the number of inequality constraints, the degree of coupling between monomials in each inequality, and the RGP approximation and uncertainty set used.

3.7.2 Effect of robustness on multiobjective performance

One of the benefits of convex and difference-of-convex optimization methods is the ability to optimize for different objectives [94]. As a demonstration, we optimize the aircraft without uncertainty for 6 different objectives, and show the non-dimensionalized figures of merit in Table 3.3. Since the model is physics based, the model can even accommodate objectives such as wing area which are often unintuitive and not considered. The resulting aircraft differ significantly with respect to certain objectives, while being similar in may others. As an example, takeoff weight for all aircraft are 0.87 to 1.22 times the baseline total fuel solution, while engine weight varies from 0.88 to 3.62 times the baseline. These demonstrate the importance of considering many objectives in design, and underline the power of SPs in helping consider the multiobjective performance of engineered systems.

We demonstrate the benefits of RSPs in multiobjective optimization by considering uncertainty while optimizing for the same objectives. We perform the optimization of the

Objective	Total fuel	Total cost	Takeoff weight	1/(Cruise L/D)	Engine weight	Wing area
Total fuel	1.00	1.00	1.00	1.00	1.00	1.00
Total cost	2.51	0.64	0.98	2.55	3.62	0.98
Takeoff weight	1.37	0.89	0.87	1.43	1.42	0.87
1/(Cruise L/D)	1.29	1.00	1.11	0.77	2.48	1.53
Engine weight	1.35	1.53	1.22	1.40	0.88	2.78
Wing area	1.37	0.89	0.87	1.43	1.43	0.87

Table 3.3: Non-dimensionalized variations in performance of aircraft optimized for different objectives. Objective values are normalized by the total fuel solution.

aircraft with no uncertainty, and both box and ellipsoidal uncertainty ($\Gamma = 1$) for the objective functions in Table 3.3, and plot the results on radar plots. Radar plots are useful because they allow engineers to visualize the performance of designs in many dimensions. One way to envision the multiobjective performance of the aircraft is to consider the area of the polygon defined by the aircraft's performance as the figure of merit; the smaller the better.

Figure 3-8 shows the effects of robustness on the different worst-case performance metrics of the different aircraft. As expected, the box uncertainty set is strictly more conservative than the ellipsoidal uncertainty set in the optimized objective. However, the tradeoffs in the other objectives are less clear. Note that the radar plots show the worst-case performance of the vehicles, although this analysis can also be performed for the mean performance of the aircraft determined through MC simulation.

This multiobjective comparison underscores the sensitivity of different objectives to level of robustness and by extension parameter uncertainty. For example, the engine weight of the 1/(Cruise L/D) solution is highly sensitive to level of robustness, whereas the engine weight of the total (time and fuel) cost aircraft is insensitive. Therefore, we might want to consider total cost to be our overall objective instead of 1/(Cruise L/D) if we are relatively averse to risk in engine versus airframe design. Robustness can affect the efficacy of different choices of objective function in ensuring multiobjective performance. Since RSPs can be solved quickly and reliably over a variety of objective functions, they allow engineers to understand these kinds of complex tradeoffs early on in the design process.

Based on these observations, we argue that there could be significant value left on the table if uncertainty is not considered with sufficient mathematical rigor in early phases of the design process. RSPs allow engineers to capture complex tradeoffs in nonlinear optimization problems while considering uncertainty, resulting in *less conservative* solutions than solutions that implement margins and other less mathematically rigorous methods for risk mitigation. Thus RSPs improve significantly on the paradigms of design under uncertainty in use in the aerospace industry today.

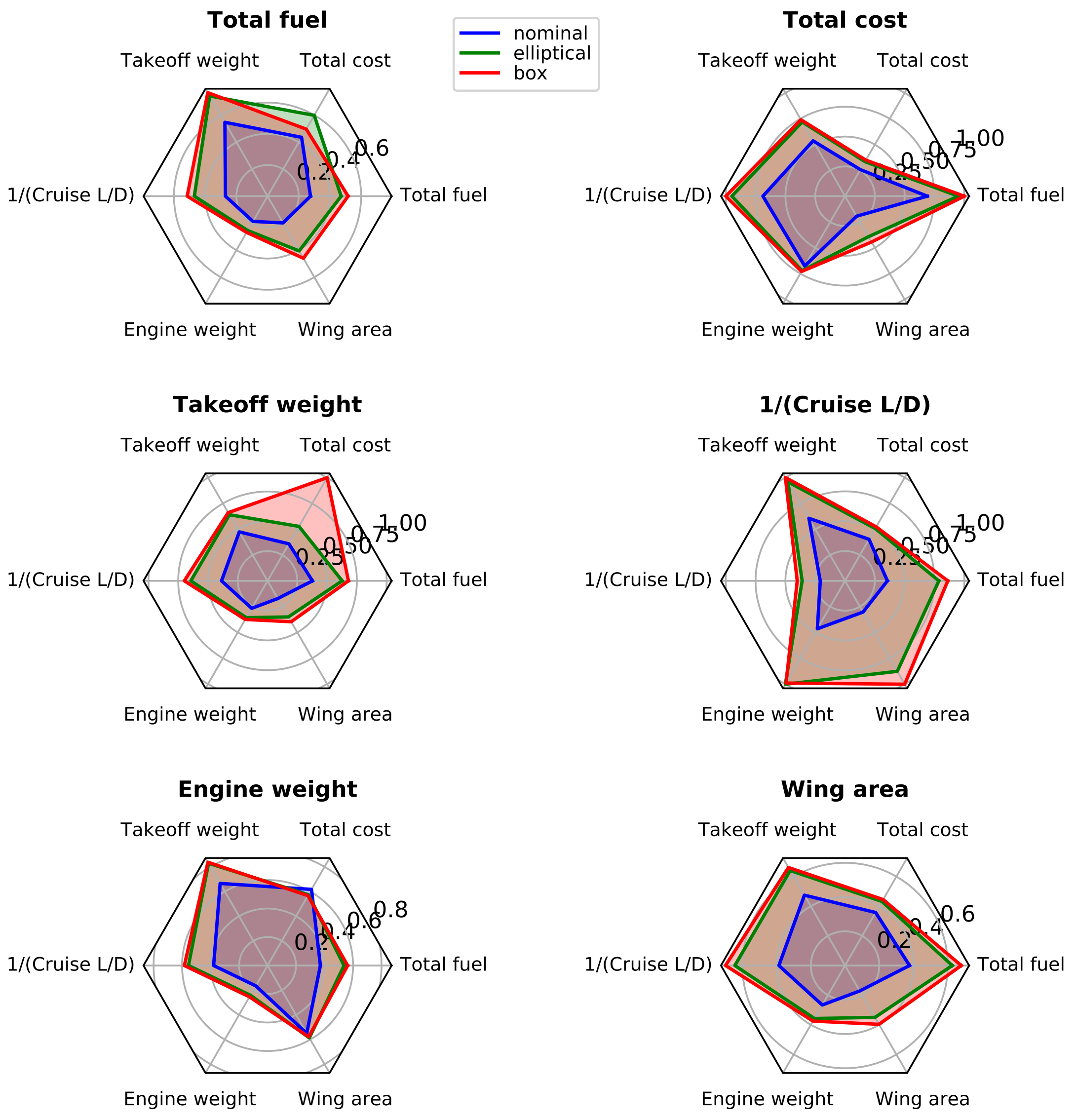
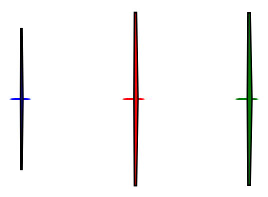
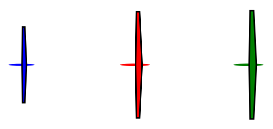


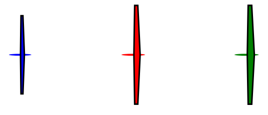
Figure 3-8: Radar plots of aircraft performance. The bolded titles are the optimized objectives for each plot, and the individual plots show the non-dimensionalized multiobjective performance of the aircraft, designed under different uncertainty sets.



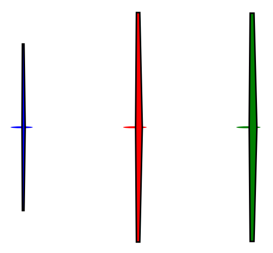
(a) Total fuel



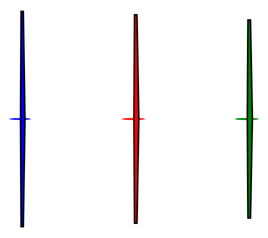
(b) Total cost



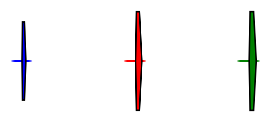
(c) Takeoff weight



(d) 1/(Cruise L/D)



(e) Engine weight



(f) Wing area

Figure 3-9: Sketches of the aircraft for radar plots in Figure 3-8. Drawn to scale for comparison.

Table 3.4: Results of original RO problem versus its goal programming counterpart in terms of size of uncertainty set Γ, objective penalty δ, and probability of failure. Both methods use the Best Pairs formulation under ellipsoidal uncertainty. The designs obtained through the two different methods match.

RO form	Γ	δ	PoF	Goal form	δ	Γ	PoF
	0.00	7.95×10^{-5}	0.88		7.95×10^{-5}	-	-
	0.10	0.0525	0.73		0.0525	0.10	0.73
	0.20	0.108	0.59		0.108	0.20	0.59
	0.30	0.168	0.40		0.168	0.30	0.39
	0.40	0.231	0.25		0.231	0.40	0.25
	0.50	0.298	0.10		0.298	0.50	0.10
	0.60	0.370	0.06		0.370	0.60	0.07
	0.70	0.447	0.03		0.447	0.70	0.03
	0.80	0.519	0.01		0.519	0.80	0.01
	0.90	0.618	0.00		0.618	0.90	0.00
	1.00	0.714	0.00		0.714	1.00	0.00

3.7.3 Risk minimization problems

All of the previous multiobjective analyses have assumed that we have an understanding of exactly the amount of uncertainty we are willing to tolerate. However, minimizing risk can also be the objective of our model. This would suggest the following formulation:

$$\begin{aligned} \max \quad & \Gamma \\ \text{s.t.} \quad & \max_{\|\mathbf{u}\| \leq \Gamma} f_i(\mathbf{x}, \mathbf{u}) \leq 0, \; i \in [n], \\ & f_0(\mathbf{x}) \leq (1+\delta) f_0^*, \; \delta \geq 0, \end{aligned} \tag{3.9}$$

where f_0^* is the optimum of the nominal problem and δ is a fractional penalty on the objective that we are willing to sacrifice for robustness, which gives $(1+\delta)f_0^*$ as the upper bound on the objective value. Intuitively, this is a form of goal programming, where we specify the exact maximum worst-case value of an objective we can tolerate with the goal of maximizing the size of the uncertainty set we can handle.

The robust goal programming formulation in (3.9) is clearly not equivalent to the robust optimization in (3.1), but should yield the same results if there is no optimality gap between the methods. To show this, we use the worst-case objective values from the PoF study shown in Figure 3-6 as the δ inputs to the goal programming model, and compare the results. The results are presented in Table 3.4. Note that the two methods were evaluated MC runs using the same 100 realizations of the uncertainty, for consistency in PoF results.

Firstly, note that there are no results reported for the goal program for zero uncertainty, $\Gamma = [0.00]$. Since the feasible set of this problem is a point design, the signomial program solution heuristic declares the problem infeasible after being unable to locate the singular feasible region. However when we positively perturb the singular δ, the goal program has a non-empty feasible set and returns the same solution as the original RO method. Otherwise, the Γ values found by the goal program match exactly with the original RO problem. We confirm that both methods produce the same designs by examining the physical dimensions of the aircraft, and through the probability of failure found through MC simulation in Table 3.4. Note that there are small discrepancies in the PoF, notably in the values for $\Gamma = [0.3, 0.6]$. This is possible because there are uncertain realizations that can fall in or out of feasibility due to numerical precision. The interior point solvers used cannot make computations exactly [70].

We can also expand this framework to perform multivariate goal programming, by changing (3.9) to include all objectives we are interested in:

$$f_{0,j}(\mathbf{x}) \leq (1+\delta_j) f^*_{0,j},\ \ \delta_j \geq 0,\ \ j \in [m]. \tag{3.10}$$

The benefit of goal programming is that it allows us to explore multidisciplinary tradeoffs without having to enumerate the design space along each objective direction. The term multiobjective optimization is misleading because you can only optimize for one objective at once. The design is going to be influenced by how engineers weigh different objectives, and it is not obvious whether an objective should be a constraint instead. The most fundamental choice that an engineer can make in design is what the objective function is, and it is often the case that there are many potential objectives that are conflicting. But risk is ubiquitous in engineering design problems, so goal programming allows risk to be used as a global design variable against which all other objectives can be weighed.

3.8 Discussion

In this section, we discuss the current RSP implementation and results, and propose areas for future work. There are a number of potential improvements to the RSP framework that would be beneficial in the context of conceptual engineering design.

3.8.1 Constraint-wise robustification

In this study, we do not discriminate between the kinds of constraints violated. However, it would be possible to rank the severity of constraint violations so as to penalize some (e.g. structural safety) more heavily than others (maximum range). This would inject further realism into design under uncertainty since some violations contribute to program risk more significantly than others.

There are new developments in RO that allow for the derivation of probabilistic guarantees of constraint violation [6, 9]. One especially relevant result from Ben-Tal and Nemirovski [6] is the proof that for a box-ellipsoidal uncertainty set $\mathcal{U} = \{\mathbf{a} : ||\mathbf{a}||_\infty \leq 1,\ ||\mathbf{a}||_2 \leq \rho\}$, the probability of constraint violation is given by

$$\mathbb{P}_{\mathbf{a}}(\mathbf{a}^T\mathbf{x} > b) \leq e^{\rho^2/2} \tag{3.11}$$

and is dimension independent. Since RGPs are a direct extension of robust linear programs into geometric space, these guarantees also hold for RGPs and are locally correct for RSPs as well. Thus we can integrate a constraint-wise robustification framework into our RSP framework so that designers can tune robustness to individual constraint violations based on probabilities of failure. This will help clarify the relationship between probability of failure and uncertainty sets in conceptual design.

3.8.2 Sensitivities to parameter uncertainties

The solution of a GP comes with the value of the dual variables at negligible computational cost. The dual variables can be used to compute the optimal sensitivities, i.e. the sensitivity of the optimal cost to changes in problem parameters, which are insightful in conceptual design. While the dual variables of the final GP iteration of the RSP solution can be used to compute the sensitivities to the means of uncertain parameters μ, it would be beneficial to replicate this post-processing step for the 3σ uncertainties as well. While it is not clear whether there is an analytical representation of these $\frac{\partial f_0^*}{\partial(3\sigma_i)}$s, it is at a minimum possible to compute these using finite differencing. The sensitivities to 3σs could inform engineers about where to concentrate efforts in order to mitigate the sensitivity of the objective function to uncertain parameters.

3.8.3 Implementation of RGPs and RSPs for novel uncertainty sets

Another potentially valuable extension to the proposed framework is the extension of RGPs and RSPs to sets other than norm sets, with the purpose of restricting uncertain outcomes further and thus reducing conservativeness. One example would be to take the intersection of the L0-norm and L2-norm sets. This method can be used to set the total size of the uncertainty set in a Euclidian sense, but then also to restrict the uncertainty to a subset of the uncertain parameters. This also turns the problem into an mixed integer robust optimization problem which poses interesting computational challenges.

3.8.4 RO for the evaluation of adaptable engineered systems

With respect to potential new applications, RO opens up the possibility to discover and analyze with mathematical rigor the benefits of adaptable architectures in engineering design versus more traditional point designs. Some examples of these are modular designs, morphing designs, adaptively manufactured designs and aircraft families. It is likely that these types of engineered robustness become more effective at reducing program risk in presence of uncertainty, since they are more likely to deliver value under adverse stochastic outcomes. RO would allow designers to more rigorously quantify the benefits of adaptable systems.

3.9 Conclusion

This chapter has motivated the use of RSPs in conceptual engineering design, in lieu of the mathematically non-rigorous methods for optimization under uncertainty widely used in the aerospace industry today. We have developed a tractable RSP formulation in response to a need to optimize over uncertain parameters, extending an existing tractable approximate RGP framework to non-log-convex problems. This RSP formulation is a valuable contribution to the fields of robust optimization and difference-of-convex programming. The Python implementation of RSPs used in this thesis is open-source and available via the link in Appendix B.1.

RSPs have a wide variety of potential applications in engineering design. We demonstrated using an unmanned aircraft design problem that the use of RSPs in conceptual design will result in systems that are more robust to uncertainties in operational parameters, such as payload mass and range, as well as uncertain environmental and manufacturing parameters. Unlike legacy methods, this robustness has probabilistic guarantees, where sets of size $\Gamma = 1$ protect against all realizations of 3σ uncertainty for a given set of parameters. Thus engineers can use robust signomial programming to trade off robustness and optimality within engineered systems in a tractable and mathematically rigorous manner.

We compared aircraft designs under fixed parameters and margins with robust designs over box and ellipsoidal uncertainty sets. We confirmed that the box design is strictly more conservative than the margin design, by simulating both designs over the same uncertain outcomes. This indicates that the traditional method of allocating margins by observing the local sensitivities of the nominal solution is inadequate, since it does not represent the worst-case outcomes of 3σ uncertain parameters as claimed. Furthermore, we showed that the box design has approximately the same expectation and standard deviation of performance as the margin design, but provides probabilistic guarantees of feasibility unlike its counterpart.

We also confirmed that the ellipsoidal design is strictly less conservative than the margin or box designs while protecting against the same parametric uncertainties. Since designs found using RSPs under ellipsoidal uncertainty are less conservative than designs found through traditional methods, RSPs have the potential to reduce the program risk and increase the performance of designs compared to traditional methods, with no sacrifice in reliability.

RO has the potential to change current aerospace design paradigms by introducing mathematical rigor to design under uncertainty. Current aerospace conceptual design practices still rely heavily on the expertise of established engineers even in absence of prior experience exploring the design space. RSPs provide new opportunities in aerospace conceptual design since they are compatible with physics based models that are deprived of or lacking in data, and bring quantitative measures of design reliability to the table. These new opportunities are discussed in greater detail in Section 4.2.

Chapter 4

Conclusion

To quote Theodore von Kármán,

"Scientists study the world as it is, engineers create the world that never was."

This thesis is inspired by the potential of cutting edge optimization methods to move us closer to that future world, where we continue to enjoy the benefits of aerospace engineering while mitigating its deleterious effects on planet Earth.

Aerospace design is a particularly multidisciplinary and multiobjective proposition that requires good conceptual design tools. Since few aerospace concepts have made it from initial sketches to final products, conceptual design tools must rely on models that capture the tradeoffs between different interacting disciplines instead of prior experience. These include physical constraints such as those in aerodynamics and structures, but also important human factors such as economics and ergonomics. In addition, these tools need to capture and protect against the uncertainties that are ubiquitous in the physical world. The methods in this thesis address these challenges and enable better conceptual design.

Conceptual design is *decision making with a knowledge of tradeoffs*, and optimization is *the mathematical language of decision making and tradeoffs.* While there is a sense that aerospace design is as much an art as it is a science, optimization is the language that engineers use to both define design problems, and also communicate decisions and tradeoffs in a quantitative manner. Optimization cannot and does not replace engineering expertise; it is a powerful tool that enables aerospace engineers to push the envelope of what is possible, by making decisions informed by engineering models, and helping communicate them to many stakeholders. In this thesis we extend the capabilities of existing design optimization tools in two fundamental ways.

The first is by expanding the scope of objectives and constraints that we can capture using optimization. In Chapter 2, we have proposed methods to effectively optimize over constraints and objective functions with arbitrary mathematical primitives. These tools give flexibility to engineers in embedding a wide variety of constraints from different disciplines into conceptual design frameworks, using mature optimization and ML methods.

The second is by rigorously considering uncertainty and associated risks in a design optimization setting. Aerospace systems are sensitive to assumptions made about technological capabilities, environmental factors, manufacturing quality and the future state of markets and regulatory agencies. The methods proposed in Chapter 3 allow engineers to consider tradeoffs in the presence of uncertainty, reducing risk and increasing confidence in both the engineered systems and design tools.

While the methods have been presented in the context of aerospace conceptual design, they are sufficiently general to be applied in all phases of the decision making process in a variety of fields. In addition, the methods consider the importance of the human in the loop; intuition, tractability and practicality have been key factors in the formulation of the methods in this thesis. We highlight our key contributions below.

4.1 Overview of contributions

Chapter 2 proposes a global optimization approach that effectively addresses a combination of explicit and inexplicit constraints in design. The method is more general than other state-of-the-art methods in the literature, only requiring that decision variables in nonlinear constraints lie in a bounded domain. Our specific contributions are as follows:

1. We propose constraint learning using optimal decision trees as a method to make efficient, optimization-compatible approximations of difficult constraints and objectives.
2. We introduce an ensemble of sampling methods for constraint learning, that balance the need for constraint learners to be both locally and globally accurate over the domain of decision variables.
3. We solve our efficient approximations of global optimization problems using commercial MIO solvers, demonstrating our method's performance over a number of benchmarks and real-world problems.

Chapter 3 proposes a RO approach for aerospace design by formulating the robust signomial program. This method improves on state-of-the-art stochastic optimization methods used in the aerospace literature by demonstrating better tractability as well as probabilistic guarantees of constraint satisfaction. We contribute to the existing literature on RO as well as in aerospace design in the following ways:

1. We propose and implement a tractable RSP formulation for optimization under uncertainty, that is sufficiently general to address aerospace design problems.
2. We address an aircraft design problem using our method, showing its practicality, tractability and ability to consider uncertainty with mathematical rigor.
3. We propose new methods to consider risk in aerospace design by leveraging the potential of RO in multiobjective optimization problems, and by developing risk-minimizing goal programming methods.

4.2 Potential future applications

While we have discussed future work for global optimization via constraint learning, and for robust optimization for engineering design in Sections 2.8 and 3.8 respectively, we now speculate as to how the proposed methods may enable new aerospace concepts through their application.

ML-driven global optimization has the potential to revolutionize the integration of data and simulations into optimization, during all phases of engineering design and at all levels of simulation fidelity. This is especially important as many future aerospace concepts have greater level of coupling between different engineering disciplines, requiring the embedding of higher fidelity tools into conceptual design. Some examples of such future concepts are blown-lift, boundary-layer-ingesting and morphing vehicles. Constraint learning in this context would enable a better understanding of fundamental design tradeoffs from experimental data, and allow for a more seamless conceptual design process.

Since the proposed MDO methods leverage mixed-integer optimization, they can efficiently include discrete decisions, which are critical for many aerospace applications. One domain of promise is in electric vehicle design, especially with distributed propulsion, where OCT-HaGOn can consider discrete decisions such as the addition of battery packs, circuitry and propulsors in a tractable framework. In addition, integer decision variables allow for aerospace design with component libraries. Component selection has been formulated in the past as a convex MIO [71]; the methods described in this thesis allow for component selection over arbitrary constraints and objectives.

Since our MDO methods allow for arbitrary constraints and objectives, they allow for efficient formulations of nonlinear dynamics and control problems as well. The importance of optimization in aerospace guidance and control problems is well documented [64]. These applications require fast on-vehicle computation, which is often achieved by imposing convexity requirements on the underlying optimal control problems. Constraint learning and efficient MIO may be a valid alternative for solving nonlinear control problems, especially those that have intractable nonlinearities. Similarly, constraint learning and MIO enables for the efficient solution of scheduling problems, while considering the nonlinearities in the operating characteristics of the agents.

A final benefit of using the constraint learning approach in aerospace design is in multiobjective optimization, and specifically in the efficient generation of Pareto fronts. This thesis has only considered benchmark and real-world problems with single objectives, but the combined MIO and PGD approach can be generalized to the multiobjective context. As discussed in Section 2.6, once the approximating OCT-Hs are trained over the difficult constraints in a design problem, the MIO and PGD steps are more efficient at generating new solutions than comparable global optimization tools. This potential can be leveraged, along with methods for Pareto-optimal solution generation such as those described by Kim and de Weck [51], to generate Pareto-optimal designs in arbitrary number of objective dimensions.

While the potential upsides of RO in aerospace are perhaps less glamorous, they are no less significant. Besides the clear benefits of RO in allowing designs with uncertainty protection, the optimal solutions of RO problems give engineers a clear sense of what uncertainties have the greatest bearing on design choices. This allows for a more strategic allocation of research and development efforts, to mitigate uncertainties and/or improve parameter values to maximize system performance.

RO of engineered systems, by optimally allocating design margins, has the ability to reduce the cost overruns of aerospace projects due to redesigns. Redesigns are most often required when constraints are violated; due to the coupled nature of optimization problems, even a single constraint violation can have system-wide consequences. RO mitigates this in two ways. First and most evidently, RO optimally allocates margin on every constraint, that makes it robust to all uncertain outcomes as defined by an uncertainty set. Additionally, using goal programming formulations, engineers can precisely quantify how the margin on constraint satisfaction changes as uncertain outcomes are realized. As the design process proceeds, the bounds on the uncertainty grow tighter, potentially affecting the level of uncertainty protection of each constraint. By computing the changes in these margins through goal programming, engineers can decide whether or not a redesign is required.

Aerospace vehicles often have many levels of redundancy in components. This is especially true in mission-critical parts; taking the example of a commercial aircraft, some examples of redundant systems are flight computers, tail structures and hydraulic systems. RO may be useful to trade-off the redundancy and resiliency of different systems, depending on uncertainties in the operational environment.

RO can also enable better evaluation of certain kinds of adaptable aerospace designs. While robust designs protect against uncertainty by optimally allocating margins to every constraint, adaptable designs add uncertainty protection by changing the design variables themselves. A simple example is a fighter aircraft that is able to increase its range by adding drop tanks under its wings. As with every added capability, adaptability comes with tradeoffs. A fighter aircraft with drop tanks will likely be heavier, and have higher drag and lower maneuvrability when fully loaded, than a fighter aircraft that had been designed to achieve the same range with internal fuel. RO would allow for a more rigorous consideration of tradeoffs in adaptable architectures, considering the variance in the types of missions addressed as well as changes in environmental factors.

When applied to aerospace concepts with particularly high levels of exposure to uncertainty, RO can inject realism into the design process. One example is in solar vehicles, which rely heavily on assumptions and projections about technological capabilities and meteorological conditions. These vehicles are often on the very edge of infeasibility even using optimistic outcomes of uncertain parameters. In this context, RO would better inform designers about the current and future feasibility of different concepts and mission architectures, without the “inherent optimism in initial concept designs due to competitive pressures” [36].

There are also design contexts in which the joint application of constraint learning with robust MIO may be the path forward. One example is in the design of aerospace systems that rely on uncertain simulations. This may be either due to the simulations being inaccurate approximations of the real world, or the simulations requiring uncertain parameter inputs. In this particular case, it may be appropriate to apply constraint learning to discover the underlying model in an optimization-compatible form. Using OCT-H learners from this thesis, robustness can be injected into the model in two ways. The first would be in the training stage, where tree training is modified so that the feasible leaf predictions are robust to perturbations of the data. The second method would be to train the tree on data without uncertainty, and then add robustness to the union-of-polyhedra approximations using techniques from robust linear programming. In both cases, constraint learning and robust optimization can be leveraged to optimize systems whose performance is described by the outcomes of simulations with uncertain inputs and/or outputs.

Appendix A

Appendices for Global Optimization via Optimal Decision Trees

A.1 OCT-HaGOn implementation

OCT-HaGOn is implemented in Julia 1.5.4, and will be available for use at `https://1ozturkbe.github.io/research`, pending submission of the method to a journal in January 2022. The current implementation requires an academic license for Interpretable AI [48], but a lightweight version without Interpretable AI is also in development. While CPLEX is OCT-HaGOn's default solver, it also supports other MIO solvers that are compatible with JuMP.jl version 0.21.5 [33].

A.2 Optimizers

In Chapter 2, we use a variety of commercially available and free solvers to address different types of optimization problems. This appendix provides a quick overview of the different optimization tools, the versions used and their capabilities as of writing, as well as their specific applications to different problems in Chapter 2.

- **CPLEX v20.1.0.0**: CPLEX, short for ILOG CPLEX Optimization Studio, is a mixed-integer convex optimizer. It is the default solver of OCT-HaGOn, since CPLEX is available for free to solve problems with up to 1000 variables and constraints. In addition, academics can get an unlimited, no-cost academic license. CPLEX is used within OCT-HaGOn to solve the tree-based MI approximations of global optimization problems, as well as the MI-quadratic optimizations required for the PGD iterations. CPLEX is also used in the machinery of BARON, another global optimizer; see below.
- **Gurobi v9.1.1**: Gurobi is a mixed-integer convex optimizer [42]. Gurobi is available at no cost via an academic license. Due to its ability to address mixed-integer bilinear optimization problems, Gurobi was used to solve the discretization of the OOS problem in Section 2.7.2, as a benchmark for OCT-HaGOn.
- **CONOPT v3.10:** CONOPT is a gradient-based nonlinear optimizer [30]. It was

used to solve two large benchmarks in Section 2.6, via a one-year demo license obtained through the General Algebraic Modeling System (GAMS) interface.

- **IPOPT v3.13.4:** IPOPT is a freely available interior point optimizer for NLPs [92]. It was used in Section 2.6 to solve two large benchmarks, and in Section 2.7.1 to address the speed reducer problem.

- **BARON v21.1.13:** BARON is a commercially available MINLP solver that accepts a subset of nonlinear primitives [81]. BARON uses CPLEX as its back-end MIO solver for its branch-and-reduce solution approach. We purchased a BARON license to be able to solve 5 small benchmarks and 2 large benchmarks in Section 2.6.

A.3 Speed reducer problem

We detail the constraints in the speed reducer problem addressed in Section 2.7.1. Note that it has been transcribed from [77] into standard form as defined in Section 2.4.1.

$$\begin{aligned}
\min_{\mathbf{x}} \quad & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
& \quad - 1.5079x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) \\
\text{s.t.} \quad & -27 + x_1x_2^2x_3 \geq 0, \\
& -397.5 + x_1x_2^2x_3^2 \geq 0, \\
& -1.93 + \frac{x_2x_6^4x_3}{x_4^3} \geq 0, \\
& -1.93 + \frac{x_2x_7^4x_3}{x_5^3} \geq 0, \\
& 110.0x_6^3 - \left(\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right)^{0.5} \geq 0, \\
& 85.0x_7^3 - \left(\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6\right)^{0.5} \geq 0, \\
& 40 - x_2x_3 \geq 0, \\
& x_1 - 5x_2 \geq 0, \\
& 12x_2 - x_1 \geq 0, \\
& x_4 - 1.5x_6 - 1.9 \geq 0, \\
& x_5 - 1.1x_7 - 1.9 \geq 0, \\
& \mathbf{x} \geq [2.6, 0.7, 17, 7.3, 7.3, 2.9, 5], \\
& \mathbf{x} \leq [3.6, 0.8, 28, 8.3, 8.3, 3.9, 5.5], \\
& x_3 \in \mathbb{Z}.
\end{aligned}$$

Iteration	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Objective
1	3.5	0.7	17.0	7.3	7.71590	3.35011	5.28718	3018.809
2	3.5	0.7	17.0	7.3	7.71590	3.35011	5.28718	2994.674
3	3.5	0.7	17.0	7.3	7.71590	3.35021	5.28718	2994.700
4	3.5	0.7	17.0	7.3	7.71532	3.35021	5.28665	2994.355
5	3.5	0.7	17.0	7.3	7.71532	3.35021	5.28665	2994.355

Table A.1: Speed reducer PGD iterations.

A.3.1 Speed reducer PGD iterations

The speed reducer problem is converged to a feasible and locally optimal solution from the MIO solution in 4 PGD steps. The decision variable and objective values at each iteration are given in Table A.1.

A.4 Satellite OOS problem

The satellite OOS problem has the following decision variables and associated dimensions, where n_s is the number of client satellites.

$$\begin{aligned}
\text{Satellite order variables} &: z_{i,j} \in \{0,1\}, && i,j \in [n_s],\\
\text{Orbit radii} &: r_{\text{orbit},i} \in [r_{\text{orbit,min}}, r_{\text{orbit,max}}], && i \in [n_s - 1],\\
\text{Orbital periods} &: T_{\text{orbit,i}} \in [T_{\text{orbit,min}}, T_{\text{orbit,max}}], && i \in [n_s - 1],\\
\text{Orbital period differences} &: \Delta T_{\text{orbit,i}} \in [\Delta T_{\text{min}}, \Delta T_{\text{max}}], && i \in [n_s - 1],\\
\text{True anomalies} &: \theta_i \in [-\pi, \pi], && i \in [n_s - 1],\\
\text{Transfer times} &: t_{\text{transfer,i}} \in [0, t_{\text{transfer,max}}], && i \in [n_s - 1],\\
\text{Maneuver times} &: t_{\text{maneuver,i}} \in \mathbb{R}^+, && i \in [n_s - 1],\\
\text{Orbital revolutions} &: N_{\text{orbit},i} \in [50, 500], && i \in [n_s - 1],\\
\text{Orbital entry mass ratios} &: f_{\text{entry,i}} \in [1, 1.0025], && i \in [n_s - 1],\\
\text{Orbital exit mass ratios} &: f_{\text{exit,i}} \in [1, 1.0025], && i \in [n_s - 1],\\
\text{Wet mass} &: m_{\text{wet}} \in [m_{\text{dry}}, 2000], &&\\
\text{Intermediate masses} &: m_{i,j} \in [m_{\text{dry}}, 2000], && i \in [n_s - 1],\ j \in [5],\\
\text{Transferred fuel masses} &: m_{\text{fuel,i}} \in [m_{\text{fuel,min}}, m_{\text{fuel,max}}], && i \in [n_s].
\end{aligned}$$

The objective function is to minimize the wet (i.e. fueled) mass of the satellite. Note that the orbital quantities define the phasing orbits that the servicer uses to transfer between client satellites, and all altitudes are converted to radii with respect to the center of the Earth for simplicity.

The bounds $r_{\text{orbit,min}}$, $r_{\text{orbit,max}}$, m_{dry} are defined in Table 2.8 as the minimum and maximum servicer altitudes, and the servicer dry mass respectively. $m_{\text{fuel,min}}$ and $m_{\text{fuel,max}}$ are the minimum and maximum of the fuel requirements shown in Figure 2-11. Since $t_{\text{maneuver,i}}$

is not in any nonlinear constraints, it doesn't require bounds. The remaining bounds are defined as a function of problem parameters such as specific impulse I_{sp}, maximum service time $t_{\max}$, client orbital altitude r_{client} and client fuel requirements $\Delta m_{\text{cf,i}},\ i \in [n_s]$; as well as physical constants such as the gravitational constant μ, and gravitational acceleration g.

$$\begin{aligned}
T_{\text{client}} &= 2\pi\sqrt{\frac{r_{\text{client}}}{\mu}} \\
T_{\text{orbit,min}} &= 2\pi\sqrt{\frac{r_{\text{orbit,min}}}{\mu}} \\
T_{\text{orbit,max}} &= 2\pi\sqrt{\frac{r_{\text{orbit,max}}}{\mu}} \\
\Delta T_{\text{orbit,min}} &= -\max(|\mathrm{T_{orbit,i}} - \mathrm{T_{client}}|,\ \forall \mathrm{i} \in [\mathrm{n_s} - 1]) \\
\Delta T_{\text{orbit,max}} &= \max(|\mathrm{T_{orbit,i}} - \mathrm{T_{client}}|,\ \forall \mathrm{i} \in [\mathrm{n_s} - 1]) \\
t_{\text{transfer,max}} &= 2\pi\sqrt{\frac{r_{\text{orbit,max}} + r_{\text{client}}}{8\mu}}
\end{aligned}$$

A.4.1 Linear constraints

The constraints are given below with brief descriptions.

$$\begin{aligned}
\text{Each client visited once} &: \sum_{i=1}^{n_s} z_{i,j} = 1, && \forall j \in [n_s] \\
\text{One refuel per rendezvous} &: \sum_{j=1}^{n_s} z_{i,j} = 1, && \forall i \in [n_s] \\
\text{Fuel required for } i\text{th client} &: m_{\text{fuel,i}} = \sum_{j=1}^{n_s} \Delta m_{\text{cf,j}} z_{i,j}, && \forall i \in [n_s] \\
\text{True anomaly from client } i \text{ to } i+1 &: \theta_i = \sum_{j=1}^{n_s} \Bigg((-\pi + 2\pi j/n_s)(z_{i+1,j} - z_{i,j}) \Bigg), && \forall i \in [n_s - 1] \\
\text{Wet mass} &: m_{\text{wet}} = m_{1,1} + m_{\text{fuel,1}} \\
\text{Intermediate fuel transfers} &: m_{i,5} = m_{i+1,1} + m_{\text{fuel,i+1}}, && \forall i \in [n_s - 2] \\
\text{Dry mass} &: m_{n_s-1,5} = m_{\text{dry}} + m_{\text{fuel,n}_\text{s}} \\
\text{Orbital period difference} &: \Delta T_{\text{orbit,i}} = T_{\text{orbit,i}} - T_{\text{client}}, && \forall i \in [n_s - 1] \\
\text{Total maneuver time} &: \sum_{i=1}^{n_s-1} t_{\text{maneuver,i}} \le t_{\max}
\end{aligned}$$

A.4.2 Nonlinear constraints

The nonlinear constraints fall into 7 distinct forms, which are repeated in the satellite dynamical system. The list of 60 nonlinear constraints, as well as brief descriptions are below:

- Transfer orbit entry burn ($n_s - 1$ constraints): Describes mass ratio (entry mass over exit mass) of the satellite during transfer orbit entry.

$$f_{\text{entry,i}} = \max\left[\exp\left(\frac{1}{gI_{\text{sp}}}\sqrt{\frac{\mu}{r_{\text{orbit,i}}}}\left(\sqrt{\frac{2r_{\text{client}}}{r_{\text{client}} + r_{\text{orbit,i}}}} - 1\right)\right),\right.$$
$$\left.\exp\left(\frac{1}{gI_{\text{sp}}}\sqrt{\frac{\mu}{r_{\text{client}}}}\left(\sqrt{\frac{2r_{\text{orbit,i}}}{r_{\text{client}} + r_{\text{orbit,i}}}} - 1\right)\right)\right],\ i \in [n_s - 1].$$

- Transfer orbit exit burn ($n_s - 1$ constraints): Describes the mass ratio (entry mass over exit mass) of the satellite during transfer orbit exit.

$$f_{\text{exit,i}} = \max\left[\exp\left(\frac{1}{gI_{\text{sp}}}\sqrt{\frac{\mu}{r_{\text{client}}}}\left(1 - \sqrt{\frac{2r_{\text{orbit,i}}}{r_{\text{client}} + r_{\text{orbit,i}}}}\right)\right),\right.$$
$$\left.\exp\left(\frac{1}{gI_{\text{sp}}}\sqrt{\frac{\mu}{r_{\text{orbit,i}}}}\left(1 - \sqrt{\frac{2r_{\text{client}}}{r_{\text{client}} + r_{\text{orbit,i}}}}\right)\right)\right],\ i \in [n_s - 1].$$

- Mass conservation ($4(n_s - 1)$ constraints): Couples the fractional change in mass of the satellite to the absolute change in mass during each burn phase.

$$\begin{aligned}
m_{i,1} &= f_{\text{entry,i}} m_{i,2}, && i \in [n_s - 1],\\
m_{i,2} &= f_{\text{exit,i}} m_{i,3}, && i \in [n_s - 1],\\
m_{i,3} &= f_{\text{exit,i}} m_{i,4}, && i \in [n_s - 1],\\
m_{i,4} &= f_{\text{entry,i}} m_{i,5}, && i \in [n_s - 1].
\end{aligned}$$

- Phasing orbit period ($n_s - 1$ constraints): Describes the period of the phasing orbit.

$$T_{\text{orbit,i}} = 2\pi\sqrt{\frac{r^3_{\text{orbit,i}}}{\mu}},\ i \in [n_s - 1].$$

- Transfer time ($n_s - 1$ constraints): Describes the Hohmann transfer time from the client to phasing orbit.

$$t_{\text{transfer,i}} = 2\pi\sqrt{\frac{(r_{\text{client}} + r_{\text{orbit,i}})^3}{8\mu}},\ i \in [n_s - 1].$$

- Number of transfer orbit revolutions ($n_s - 1$ constraints): Describes the number of revolutions in phasing orbit.

$$N_{\text{orbit,i}} \Delta T_{\text{orbit,i}} = T_{\text{client,i}} \theta_i,\ i \in [n_s - 1].$$

- Maneuver time ($n_s - 1$ constraints): Describes the maneuver time (transfer and phasing time) between clients.

$$t_{\text{maneuver,i}} = t_{\text{transfer,i}} + N_{\text{orbit,i}} T_{\text{orbit,i}},\ i \in [n_s - 1].$$

Appendix B

Appendices for Optimal Engineering Design Decisions Under Uncertainty

B.1 RSP implementation

Robust geometric and signomial programming formulations from [80] and this thesis are implemented in Python 3.9, and are available for use at
`https://1ozturkbe.github.io/research`. The current implementation couples to optimization models built via GPkit, a Python package that provides abstractions for using GPs and SPs in engineering design [18]. While the RSP models can be solved using CVXOPT [89], a freely available conic solver that is installed with GPkit by default, we recommend Mosek [2] due to its higher performance. Mosek is available at no cost via an academic license.

B.2 Review of robust linear programming

Principles from robust linear programming are key for approximating robust geometric programs.

Consider the system of linear constraints

$$\mathbb{A}\mathbf{x} + \mathbf{b} \leq 0$$

where

$$\begin{aligned} &\mathbb{A} \text{ is } m \times n \\ &\mathbf{x} \text{ is } n \times 1 \\ &\mathbf{b} \text{ is } m \times 1 \end{aligned} \tag{B.1}$$

where the uncertain data is contained in a set defined by (3.3) and (3.4).

B.2.1 Box uncertainty set

If the perturbation set $\mathcal{Z}$ given in (3.4) is a box uncertainty set, i.e. $\|\zeta\|_\infty \leq \Gamma$, then the robust formulation of the ith constraint is equivalent to

$$\Gamma \textstyle\sum_{l=1}^{L} |-b_i^l - \mathbf{a}_i^l \mathbf{x}| + \mathbf{a}_i^0 \mathbf{x} + b_i^0 \leq 0 \tag{B.2}$$

If only b is uncertain, i.e. $A^l = 0,\ \forall l \in [L]$, then constraint (B.2) becomes

$$\textstyle\sum_{l=1}^{L} \mathbf{a}_i^0 \mathbf{x} + b_i^0 + \Gamma \sum_{l=1}^{L} |b_i^l| \leq 0 \tag{B.3}$$

which is a linear constraint.

On the other hand, if A is uncertain, then (B.2) is equivalent to the following set of linear constraints:

$$\begin{aligned} \Gamma \textstyle\sum_{l=1}^{L} w_i^l + \mathbf{a}_i^0 \mathbf{x} + b_i^0 \leq 0 & \\ -b_i^l - \mathbf{a}_i^l \mathbf{x} \leq w_i^l,\ \forall l \in [L], & \\ b_i^l + \mathbf{a}_i^l \mathbf{x} \leq w_i^l,\ \forall l \in [L]. & \end{aligned} \tag{B.4}$$

B.2.2 Ellipsoidal uncertainty set

If the perturbation set $\mathcal{Z}$ is an ellipsoidal, i.e. $\sum_{l=1}^{L} \frac{\zeta_l^2}{\sigma_l^2} \leq \Gamma^2$, then the robust formulation of the i^{th} constraint is equivalent to

$$\Gamma \sqrt{\textstyle\sum_{l=1}^{L} \sigma_l^2 (-b_i^l - \mathbf{a}_i^l \mathbf{x})^2} + \mathbf{a}_i^0 \mathbf{x} + b_i^0 \leq 0, \tag{B.5}$$

which is a second order conic constraint.

If only b is uncertain, i.e. $\mathbb{A}^l = 0,\ \forall l \in [L]$, then (B.5) becomes

$$\textstyle\sum_{l=1}^{L} \mathbf{a}_i^0 \mathbf{x} + b_i^0 + \Gamma \sqrt{\sum_{l=1}^{L} \sigma_l^2 (b_i^l)^2} \leq 0, \tag{B.6}$$

which is a linear constraint.

B.2.3 Norm-1 uncertainty set

If the perturbation set represented by $\mathcal{Z}$ is a norm-1 uncertainty set, i.e. $\|\zeta\|_1 \leq \Gamma$, then the robust constraint is

$$\textstyle\sum_{l=1}^{L} \mathbf{a}_i^0 \mathbf{x} + b_i^0 + \Gamma \max_{l \in [L]} |b_i^l| \leq 0 \tag{B.7}$$

when $\mathbb{A}^l = 0$, and

$$\begin{aligned} \Gamma w_i + \mathbf{a}_i^0 \mathbf{x} + b_i^0 \leq 0, & \\ -b_i^l - \mathbf{a}_i^l \mathbf{x} \leq w_i,\ \forall l \in [L], & \\ b_i^l + \mathbf{a}_i^l \mathbf{x} \leq w_i,\ \forall l \in [L], & \end{aligned} \tag{B.8}$$

if $\mathbb{A}^l \neq 0$. Note that for this type of uncertainty, the robust constraints are linear.

Bibliography

[1] Jeremy Agte, Olivier de Weck, Jaroslaw Sobieszczanski-Sobieski, Paul Arendsen, Alan Morris, and Martin Spieck. MDO: Assessment and direction for advancement-an opinion of one international group. *Structural and Multidisciplinary Optimization*, 40(1-6):17–33, 2010.

[2] MOSEK ApS. *MOSEK Optimizer API for Python*, 2020. Version 9.0.

[3] S. J. Bates, J. Sienz, and D. S. Langley. Formulation of the Audze-Eglais Uniform Latin Hypercube design of experiments. *Advances in Engineering Software*, 34(8):493–506, 2003.

[4] Stuart J. Bates, Johann Sienz, and Vassili V. Toropov. Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm. *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 7(April 2004):5217–5223, 2004.

[5] R. Belie. Non-technical barriers to multidisciplinary optimization in the aerospace industry. *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, (September):1–6, 2002.

[6] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of Linear Programming problems contaminated with uncertain data. *Mathematical Programming, Series B*, 88(3):411–424, 2000.

[7] María Lorena Bergamini, Ignacio Grossmann, Nicolás Scenna, and Pío Aguirre. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Computers and Chemical Engineering*, 32(3):477–493, 2008.

[8] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and Applications of Robust Optimization. *Society for Industrial and Applied Mathematics*, 2011.

[9] Dimitris Bertsimas, Dick den Hertog, and Jean Pauphilet. Probabilistic Guarantees in Robust Optimization. *SIAM Journal on Optimization*, 31(4):2893–2920, 2021.

[10] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[11] Dimitris Bertsimas and Jack Dunn. *Machine Learning Under a Modern Optimization Lens*. Dynamic Ideas LLC, 2019.

[12] Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *Mathematical Programming*, 167(2):235–292, 2018.

[13] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, 2004.

[14] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[15] Max Biggs and Rim Hariss. Optimizing Objective Functions Determined from Random Forests. *SSRN Electronic Journal*, pages 1–46, 2017.

[16] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127, Oct 2007.

[17] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. Classification and regression trees. wadsworth int. *Group*, 37(15):237–251, 1984.

[18] Edward Burnell, Nicole B Damen, and Warren Hoburg. GPkit: a Human-Centered Approach to Convex Optimization in Engineering Design. *Proceedings of the Computer Human Interface (CHI) Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[19] Edward Burnell, Nicole B Damen, and Warren Hoburg. GPkit: A human-centered approach to convex optimization in engineering design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[20] Michael Burton and Warren Hoburg. Solar and Gas Powered Long-Endurance Unmanned Aircraft Sizing via Geometric Programming. *Journal of Aircraft*, 55(1), 2018.

[21] Michael R. Bussieck, Arne Stolbjerg Drud, and Alexander Meeraus. MINLPLib - A collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.

[22] Andre Chassein and Marc Goerigk. Robust geometric programming is co-np hard. 2014.

[23] Xin Chen, Melvyn Sim, and Peng Sun. A Robust Optimization Perspective on Stochastic Programming. *Operations Research*, 55(6):1058–1071, 2007.

[24] A. J. Conejo, F. J. Nogales, and F. J. Prieto. A decomposition procedure based on approximate Newton directions. *Mathematical Programming, Series B*, 93(3):495–515, 2002.

[25] Corinna Cortes, L.D. Jackel, and Wan-Ping Chiang. Limits on Learning Machine Accuracy Imposed by Data Quality. *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Knowledge Discovery and Data Mining*, pages 57–62, 1995.

[26] Urmila Diwekar. *Introduction to applied optimization*, volume 22. Springer Science & Business Media, 2008.

[27] Mark Drela. XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils. *Proceedings of the Conference on Low Reynolds Number Aerodynamics, Notre Dame, Indiana*, pages 1–12, 1989.

[28] Mark Drela. Pros and Cons of Airfoil Optimization. *Proceedings of Frontiers of Computational Fluid Dynamics*, (November):1–19, 1998.

[29] Mark Drela. Low-Order Modeling for Conceptual Aircraft Design and Development of the D8 Transport Concept. *Stanford AA295 Seminar*, pages 1–77, 2011.

[30] Arne S. Drud. CONOPT - A Large Scale GRG Code. *ORSA Journal on Computing*, 6(2), 1994.

[31] R.J Duffin, E.L. Peterson, and C. Zener. *Geometric programming: theory and application.* Wiley New York, 1967.

[32] Jack William Dunn. *Optimal Trees for Prediction and Prescription.* PhD thesis, Massachusetts Institute of Technology, 2014.

[33] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[34] Marco A. Duran and Ignacio E. Grossman. An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs. *Mathematical Programming*, 36:307–339, 1986.

[35] Peter I Frazier. Bayesian Optimization. *INFORMS TutORials in Operations Research*, pages 9–11, October 2018.

[36] Claude W Freaner, Robert E Bitten, David A Bearden, and Debra Emmons. An Assessment of the Inherent Optimism in Early Conceptual Designs. *Proceedings of the SSCAG / SCAF / EACE 2008 Joint International Conference*, 2008.

[37] François Gallard, Matthieu Meaux, Marc Montagnac, and Bijan Mohammadi. Aerodynamic aircraft design for mission performance by multipoint optimization. *21st AIAA Computational Fluid Dynamics Conference*, pages 1–17, 2013.

[38] Claudio Gambella, Bissan Ghaddar, and Joe Naoum-Sawaya. Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, 2021.

[39] Michael Gastegger, Jörg Behler, and Philipp Marquetand. Machine learning molecular dynamics for the simulation of infrared spectra. *Chemical Science*, 8(10):6924–6935, 2017.

[40] Juan S. Giraldo, Jhon A. Castrillon, Juan Camilo Lopez, Marcos J. Rider, and Carlos A. Castro. Microgrids Energy Management Using Robust Convex Programming. *IEEE Transactions on Smart Grid*, 10(4):4520–4530, 2019.

[41] Jan Golinski. Optimal Synthesis Problems Solved by Means of Nonlinear Programming and Random Methods. *Journal of Mechanisms*, 5(March 1969):287–309, 1970.

[42] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[43] Julia L. Higle and Suvrajeet Sen. Stochastic Decomposition: An Algorithm for Two-Stage Linear Programs with Recourse. *Mathematics of Operations Research*, 16(3):650–669, 1991.

[44] Warren Hoburg, Philippe Kirschen, and Pieter Abbeel. Data fitting with geometric-programming-compatible softmax functions. *Optimization and Engineering*, 17(4):897–918, 2016.

[45] Warren Woodrow Hoburg. *Aircraft Design Optimization as a Geometric Program*. PhD thesis, Massachusetts Institute of Technology, 2013.

[46] R. Horst, Ng.V. Thoai, and H. Tuy. On an Outer Approximation Concept in Global Optimization. *Optimization*, 20(3):255–264, 1989.

[47] Kan Lin Hsiung, Seung Jean Kim, and Stephen Boyd. Tractable approximate robust geometric programming. *Optimization and Engineering*, 9(2):95–118, 2008.

[48] Interpretable AI, LLC. Interpretable AI Documentation, 2022.

[49] P Kall and D Stoyan. Solving stochastic programming problems with recourse including error bounds. *Math. Operationsforsch. Statist. Ser. Optim.*, 13(3):431–447, 1982.

[50] Michael G Kapteyn, David J Knezevic, and Karen Willcox. Toward predictive digital twins via component-based reduced-order models and interpretable machine learning. *Proceedings of the AIAA Scitech 2020 Forum*, 2020.

[51] I. Y. Kim and O. L. de Weck. Adaptive weighted sum method for multiobjective optimization: A new method for Pareto front generation. *Structural and Multidisciplinary Optimization*, 31(2):105–116, 2006.

[52] Philippe G. Kirschen, Edward Burnell, and Warren Hoburg. Signomial programming models for aircraft design. *54th AIAA Aerospace Sciences Meeting*, Feb 2016.

[53] Philippe G. Kirschen and Warren W. Hoburg. The Power of Log Transformation: A Comparison of Geometric and Signomial Programming with General Nonlinear Programming Techniques for Aircraft Design Optimization. *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

[54] Philippe G. Kirschen, Martin A. York, Berk Öztürk, and Warren W. Hoburg. Application of signomial programming to aircraft design. *Journal of Aircraft*, 55(3):965–987, 2018.

[55] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning – accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118, 2021.

[56] Rhea P. Liem, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Multimission Aircraft Fuel-Burn Minimization via Multipoint Aerostructural Optimization. *AIAA Journal*, 53(1):104–122, 2015.

[57] Rhea P. Liem, Joaquim R.R.A. Martins, and Gaetan K.W. Kenway. Expected drag minimization for aerodynamic design optimization based on aircraft operational data. *Aerospace Science and Technology*, 63:344–362, 2017.

[58] Ming Hua Lin and Jung Fa Tsai. Range reduction techniques for improving computational efficiency in global optimization of signomial geometric programming problems. *European Journal of Operational Research*, 216(1):17–25, 2012.

[59] Thomas Lipp and Stephen Boyd. Variations and extension of the convex – concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[60] Zhi Quan Luo and Wei Yu. An introduction to convex optimization for communications and signal processing. *IEEE Journal on Selected Areas in Communications*, 24(8):1426–1438, 2006.

[61] Michael Luu and Daniel Hastings. Valuation of On-Orbit Servicing in Proliferated Low-Earth Orbit Constellations. *Proceedings of AIAA ASCEND 2020*, pages 0–14, 2020.

[62] Alessandro Magnani and Stephen P. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1):1–17, 2009.

[63] Alessandro Magnani, Sanjay Lall, and Stephen Boyd. Tractable fitting with convex polynomials via sum-of-squares. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:1672–1677, 2005.

[64] Yuanqi Mao, Michael Szmuk, and Behcet Açikmeşe. A Tutorial on Real-time Convex Optimization Based Guidance and Control for Aerospace Applications. *Proceedings of the American Control Conference*, 2018-June:2410–2416, 2018.

[65] Donato Maragno, Holly Wiberg, Dimitris Bertsimas, S. Ilker Birbil, Dick den Hertog, and Adejuyigbe Fajemisin. Mixed-Integer Optimization with Constraint Learning. *ArXiv*, pages 1–48, 2021.

[66] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary Design Optimization: A Survey of Architectures. *AIAA Journal*, 51(9):2049–2075, 2013.

[67] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[68] Velibor V. Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

[69] Tobias Morawietz and Nongnuch Artrith. Machine learning-accelerated quantum mechanics-based atomistic simulations for industrial applications. *Journal of Computer-Aided Molecular Design*, 35(4):557–586, 2021.

[70] Yurii Nesterov and Arkadi Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming.* Society for Industrial and Applied Mathematics, 1994.

[71] Johannes Norheim. Satellite Component Selection with Mixed Integer Nonlinear Programming. *IEEE Aerospace Conference Proceedings*, 2020.

[72] Berk Öztürk. Conceptual Engineering Design and Optimization Methodologies using Geometric Programming. Master's thesis, Massachusetts Institute of Technology, 2018.

[73] Berk Öztürk, Michael J. Burton, and Warren W. Hoburg. Design of an Unmanned Aerial Vehicle for Long-Endurance Communication Support. *AIAA Aviation Forum, 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, (June):1–18, 2017.

[74] Berk Öztürk and Ali Saab. Optimal aircraft design decisions under uncertainty using robust signomial programming. *AIAA Journal*, 59(5):1773–1785, 2021.

[75] Athanasios Papageorgiou, Mehdi Tarkian, Kristian Amadori, and Johan Ölvander. Multidisciplinary design optimization of aerial vehicles: A review of recent advancements. *International Journal of Aerospace Engineering*, 2018, 2018.

[76] M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991.

[77] Tapabrata Ray. Golinski's speed reducer problem revisited. *AIAA Journal*, 41(3):556–558, 2003.

[78] D.P. Raymer, American Institute of Aeronautics, and Astronautics. *Aircraft design: a conceptual approach.* Educ Series. American Institute of Aeronautics and Astronautics, 4th edition, 1989.

[79] Hong S. Ryoo and Nikolaos V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996.

[80] Ali Saab, Edward Burnell, and Warren W. Hoburg. Robust Designs via Geometric Programming. pages 1–23, 2018.

[81] N. V. Sahinidis. *BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017.

[82] Nikolaos V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.

[83] M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170, 1987.

[84] D.B. Shmoys and C. Swamy. Stochastic Optimization is (Almost) as easy as Deterministic Optimization. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[85] Jaroslaw Sobieszczanski-Sobieski and Raphael T. Haftka. Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments. *34th Aerospace Sciences Meeting and Exhibit*, 1995.

[86] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A Survey of Optimization Methods from a Machine Learning Perspective. *IEEE Transactions on Cybernetics*, 50(8):3668–3681, 2020.

[87] Gene A. Tagliarini, J. Fury Christ, and W. Page, Edward. Optimization Using Neural Networks. *IEEE Transactions on Computers*, 40(12):1347–1358, 1991.

[88] Tony Tao. *Design, Optimization, and Performance of an Adaptable Aircraft Manufacturing Architecture.* PhD thesis, Massachusetts Institute of Technology, 2018.

[89] L Vandenberghe. The CVXOPT linear and quadratic cone program solvers. 2010.

[90] Arun Verma. An introduction to automatic differentiation. *Current Science*, 78(7):804–807, 2000.

[91] Juan Pablo Vielma. Mixed Integer Linear Programming Formulation Techniques. *SIAM Review*, 57(1):3–57, 2015.

[92] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106:25–57, 2006.

[93] Wen Yao, Xiaoqian Chen, Wencai Luo, Michel van Tooren, and Jian Guo. Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles. *Progress in Aerospace Sciences*, 47(6):450 – 479, 2011.

[94] Martin A York, Berk Öztürk, Edward Burnell, and Warren W Hoburg. Efficient aircraft multidisciplinary design optimization and sensitivity analysis via signomial programming. *AIAA Journal*, pages 1–16, 2018.

[95] T. a. Zang, M. J. Hemsch, M. W. Hilburger, S. P. Kenny, J. M. Luckring, P. Maghami, S. L. Padula, and W. J. Stroud. Needs and opportunities for uncertainty-based multidisciplinary design methods for aerospace vehicles. *Nasa Tm*, 211462(July), 2002.