Discrete Optimization Models and Techniques for Problems in Data-Driven Prescriptive Analytics

**A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY**

Jongeun Kim

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY**

Prof. Jean-Philippe P. Richard

May, 2022

Acknowledgements

First and foremost I am extremely grateful to my advisors Prof. Jean-Philippe P. Richard and Prof. Mohit Tawarmalani for their invaluable advice, continuous support, and patience during my Ph.D. study. I would like to express my best gratitude to Prof. Richard for his contributions of time, advice, funding, and opportunities that made my Ph.D. experience rewarding and enjoyable. I also would like to show my appreciation to my co-advisor, Prof. Tawarmalani for the thoughtful and constructive advice and critiques. It has been my great honor to be a Ph.D. student under their guidance. Their enormous knowledge and plentiful experience have inspired me all the time in my academic research and daily life.

In addition to my advisors, I thank three other thesis committee members, Prof. Shuzhong Zhang, Prof. Zizhou Wang, and Prof. Qi Zhang for their comments and questions on the thesis.

I would like to acknowledge my research collaborators. My thanks go to Prof. Vladimir Boginski, Prof. Oleg A. Prokopyev, and Prof. Alexander Veremyev for their help and guidance on my first research paper. I would like to thank Dr. Sven Leyffer and Dr. Prasanna Balaprakash for giving me a research opportunity at Argonne National Laboratory and introducing me to the topic of symbolic regression.

I would like to express my appreciation to my friends and colleagues: Vladimir Stozhkov, Seonho Park, Bijan Taslimi, Charles Hernandez, Xiaojie Wang, Arsenios Tsokas, Jiali Huang, Zhenhuan Zhang, Jing Gao, Andy Yoon, Yu Fu, Ash Omidvar, Ahmet Tuysuzoglu, and many others, for their company and their help in research, career choices, and personal life.

Finally, I am grateful to Hyunjin, Leah, Liel, and my parents for their immense support on the long road to a Ph.D.

Abstract

In this thesis, we consider situations when optimization problems include objective and/or constraint functions whose explicit forms are not directly available. We focus on two complementary situations, when those unknown functions are described by pre-trained tree ensembles and when we train an explicit functional form using sample data points to substitute the unknown functions.

We first study optimization problems where some objective and/or constraint functions are described by pre-trained tree ensembles. This problem is known as *tree ensemble optimization*. We establish a connection between tree ensemble optimization and multilinear optimization. We develop new polyhedral results for one problem through the lens of the other problem, and vice versa. Computational experiments show that our formulations provide tighter relaxations and improve the solution times compared to formulations based on modeling piecewise-linear functions and compared to existing tree ensemble formulations.

We next study piecewise polyhedral relaxations of multilinear optimization. The developed results can be applied to model optimization problems when some objective and/or constraint functions are generalized decision trees. A *generalized decision tree* consists of a decision tree and a nonlinear model for each leaf. Given an input value, its prediction is computed by the nonlinear model associated with the leaf that the point corresponds to. We provide the first locally ideal formulation for the relaxation problem and further improve formulations by introducing linking constraints that help relate model components that are commonly treated independently in the literature. The implementation of our results inside of an open source mixed-integer nonlinear programming (MINLP) solver reduces by factor of approximately 3 the number of instances that the algorithm cannot solve within an hour with its default setting.

Finally, we study *symbolic regression*, which is a form of regression that learns functional expressions from observational data. An expression tree is typically used to represent a solution function, which is determined by assigning operators and operands to the nodes. The methodology we develop is based on MINLP, which seeks global optimality given a restriction on the size of a solution expression tree. We develop tighter

MINLP formulations and a heuristic that iteratively builds an expression tree by solving a restricted MINLP. In computational experiments that are aimed at discovering physics formula from sample data points, our methods outperform the best known approaches from the literature.

Contents

Acknowledgements **i**

Abstract **ii**

Contents **iv**

List of Tables **vii**

List of Figures **ix**

1 Introduction **1**

2 A Reciprocity Between Tree Ensemble Optimization and Multilinear Optimization **5**

2.1 Preface 5

2.2 Introduction 6

2.3 Literature Review 9

2.4 Sources of Reciprocity Between the Two Problems 11

2.4.1 TEO as a Multilinear Problem 12

2.4.2 Multilinear Optimization Over the Cartesian Product of Simplices 13

2.4.3 MOCPS as TEO 14

2.4.4 Supplements for Section 2.4 17

2.5 Facial Decomposition and its Convexification Properties 17

2.5.1 Partition Indicator Hull of a Facial Decomposition 19

2.5.2 Completability: Construction of a Facial Decomposition 25

2.5.3 Decomposability 30
2.5.4 Supplements: Proofs of Statements in Section 2.5 33
2.5.5 Supplements: Polynomial Separation Algorithm 44
2.5.6 Supplements: Connecting our Framework to the Literature . . . 45
2.6 Applications 53
2.6.1 Improved Formulations for TEO 53
2.6.2 Constrained TEO 55
2.6.3 Generalized Decision Trees 60
2.6.4 Supplements: Generation of Multi-Commodity Flow Problem Instances 63
2.6.5 Supplements: Formulations Used in the Experiments 64
2.6.6 Supplements: Unique Multilinear Function Over a Hyper-Rectangle 68
2.7 Conclusion 69

3 Piecewise Polyhedral Relaxations of Multilinear Optimization 71
3.1 Preface 71
3.2 Introduction 72
3.3 Mixed-Integer Linear Formulations Over RHPs 74
3.3.1 A Locally Ideal MILP Formulation for PPR Over RHPs 75
3.3.2 Linking Constraints 80
3.3.3 Computational Results Using Linking Constraints 86
3.4 Mixed-integer Linear Formulations Over Non-regular Partitions 88
3.4.1 Example of an Economical Non-Regular Partition 89
3.4.2 MILP formulations Over the Union of Polytopes 91
3.4.3 MILP Formulations Over Non-Regular Partitions 93
3.4.4 Computational Experiments 96
3.5 Conclusion 99
3.6 Supplements: A Formulation Used for Experiments in Section 3.4.4 . . . 100

4 Learning Symbolic Expressions: Mixed-Integer Formulations, Cuts, and Heuristics 102
4.1 Preface 102
4.2 Introduction 103

4.2.1 Review of Expression Trees 104
4.2.2 Methods, Test Problems, and Challenges for Symbolic Regression 105
4.3 An Improved MINLP Formulation of Symbolic Regression 108
4.3.1 Notation 108
4.3.2 Inputs 109
4.3.3 Decision Variables and Individual Variable Restrictions 109
4.3.4 Objective Function 110
4.3.5 Constraints 110
4.3.6 Summary 120
4.4 Sequential Tree Construction Heuristic 121
4.4.1 Definition of Neighbors and a Basic Heuristic 122
4.4.2 Heuristics to Speed Up Ideas and STreCH 123
4.5 Computational Experiments 125
4.5.1 Comparison of MINLP Formulations 127
4.5.2 STreCH and AI Feynman 129
4.6 Conclusion 132
4.7 Supplements: the MINLP Formulation Proposed in Cozad and Sahinidis (2018) 133
4.7.1 Tree-Defining Constraints 133
4.7.2 Value-Defining Constraints 133
4.7.3 Redundancy-Eliminating Constraints 135
4.7.4 Symmetry-Breaking Constraints 136
4.8 Supplements: Proof of Lemma 10 136

5 Summary and Conclusions **138**

References **140**

List of Tables

2.1 Relative LP gap of formulation (b) which is (2) in Mišić (2020) with our formulation (2.25) denoted as (a). 55

2.2 Size and tightness of formulations for a single decision tree with n input variables and m leaves. Columns `nBinVar`, `nConVar`, and `nConstr` display the numbers of binary variables, continuous variables, and constraints, respectively. Column `Integrality` indicates whether every extreme point of the LP relaxation is integral or not. 56

2.3 Average relative LP gaps for N multi-commodity transportation problem instances with (TEO) as the basis for comparison. The smallest average value(s) are in bold for each row. 57

2.4 Averages solution times for N multi-commodity transportation problem instances. The number of instances that reach one hour time limit is given as superscript. The smallest average value(s) are in bold for each row. . 59

2.5 Average relative LP gaps (with (TEO) as the basis for comparison) and solution times for N multi-commodity transportation problem instances. The number of instances that reach the three hours time limit is given as superscript. The smallest relative LP gap and average value are in bold for each row. 59

2.6 Test errors of (α, β)-ML-tree models with $(\alpha, \beta) \in \{(\infty, 0), (0, \infty), (0.1, 0.1)\}$. Column $h(\boldsymbol{x})$ displays the nonlinear functions used in generating the data set. The second row displays parameters (α, β) used for training. 63

3.1 Tightness of the first relaxation models. Each row displays the result on N instances with same maximum degree (d). N_{solved} is the number of instances for which both algorithms (with/without linking constraints) solve the first relaxation within the time limit. N_{new} is the number of instances that the first relaxation model with linking constraints closes the gap. N_{Alpine} is the number of instances that the first relaxation model without linking constraints closes the gap. GapClosed is the average of gaps closed among instances that are solved and ALPINE does not close the gap. 87
3.2 Average solution times of N instances with same maximum degree (d). . 87
3.3 Solution times and numbers of hyper-rectangles in regular/non-regular partitions for TEO instances with n input variables and T trees with maximum depth D. 98
4.1 Redundancy removing constraints. 115
4.2 Summary of the formulation and contributions compared with the benchmark formulation Cozad and Sahinidis (2018). 120
4.3 Distribution of the required depth to represent a formula in the test problems. 126
4.4 Formulations used in the experiments. 127
4.5 Required depth to represent a formula. 130
4.6 Progress of STreCH to discover $\frac{(h/(2\pi))^2}{2E_n d^2}$. 131

List of Figures

2.1 A decision-tree (a) and its hyper-rectangular partition (b). The indicator of leaf 1 is $x_{1,1}x_{2,1}$. 8

2.2 Illustration of Example 1. 13

2.3 Decision tree associated with the face F where $\mathbb{1}_F(\boldsymbol{x}) = (x_{11}+x_{13})x_{22}(x_{42}+x_{44})$ with coefficient c_F. 16

2.4 Transportation network corresponding to TP(1) for the example of Figure 2.2. 20

2.5 Grid representations for Example 1 before and after reordering of variables. 22

2.6 Schematic illustration of relationships between results. 23

2.7 Facial decomposition that is not decision-tree representable. There is no way to partition $\{y_1, \dots, y_5\}$ into two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ such that $\sum_{F \in \mathcal{F}_i} y_F$ corresponds to a face for $i = 1, 2$. 25

2.8 Refinement and completion, using a grid representation. 26

2.9 Performance profiles of solution times. The horizontal axis represents the solution time and the vertical axis represents the fraction of instances solved. Figure 2.9a displays the performance profile of 200 instances where $C \in \{2,3\}$, $D \in \{4,5\}$, and $T \in \{1,3,5,7,9\}$. Figure 2.9b displays the performance profile of 20 hard instances where $C \in \{2,3\}$, $D = 6$, and $T = 20$. 60

3.1 Performance profile of solution times. 88

3.2 Example partition (a), regular refinement (b), and non-regular refinement (c) when $n = 3$ and $K = 4$. 89

4.1 An expression tree of the pendulum formula, $T = 2\pi\sqrt{\frac{L}{g}}$. 104

4.2 Two equivalent expression trees corresponding to x_1. Both are feasible in the Cozad and Sahinidis (2018)'s formulation, while only (a) is feasible in our improved formulation. 111
4.3 Two equivalent expression trees with addition and subtraction. Both are feasible in Cozad and Sahinidis (2018), but only (a) is feasible in (4.10a). 116
4.4 Two equivalent expression trees with multiplication and division. Both are feasible in Cozad and Sahinidis (2018), but only (a) is feasible in (4.10b). 116
4.5 Invalid expression trees because of domain restriction. 118
4.6 Illustration of $\frac{1-x+x^2}{1+x}$ and $\frac{1-x}{1+x}$. 121
4.7 Two expression trees with distance three. 122
4.8 Comparison of solution times of MINLP solves with the improved formulations and those of Cozad and Sahinidis (2018). 128
4.9 Comparison of the number of nodes in the branch-and-bound tree of MINLP solves with the improved formulations and the the formulations of Cozad and Sahinidis (2018). 130
4.10 Comparison of the testing errors achieved by the STreCH approach, the MINLP approach, and AI Feynman. 132

Chapter 1

Introduction

Optimization models have been successfully used in the solution of many practical problems. Traditionally, these models take the form of optimizing an objective function, subject to functional constraints, all of which are known in closed-form. In many engineering settings, such closed-form expressions are not directly available. Rather, the values of objective and constraints might only be known at various sample points of the domain. When faced with these difficulties, one possible approach is (i) to fit functional forms, using the data available, for objective and constraints, and then (ii) to develop solution approaches and models for the resulting optimization problem. In this thesis, we tackle both aspects of this approach by developing discrete optimization methodologies for the problem of deriving best fitting functions using symbolic regression and by developing structural results to improve the solution of the nonconvex discrete optimization problems that arise when modeling constraints or objective functions through tree ensembles.

Optimization models where functions are not known explicitly frequently arise in domains as diverse as chemical engineering (Gross and Roosen, 1998), fluid mechanics (Lehnhäuser and Schäfer, 2005), and marketing (Ferreira et al., 2016). In these application areas, the resulting problems are often solved using *black-box optimization (BBO)* or *derivative-free optimization (DFO)*; see Pardalos et al. (2021); Ploskas and Sahinidis

(2021) for details. A common characteristic of BBO and DFO problems is that black-box engines are available to compute desired values, *i.e.*, the function values are obtainable (sometimes noisily) by solving computationally expensive models. In this thesis, we consider situations where the functional values are only available at certain predetermined points, *i.e.*, no black-box engine can be used to acquire the values at other points. We focus on developing solution methodologies for optimization models related to either (i) or (ii).

The choice of predictive models approximating unknown functions in step (ii) has been often limited to relatively structured functions (including convex functions) to ensure the tractability of the resulting optimization problems. Cohen et al. (2017) use linear regression to train a demand prediction model in order to design optimal sales promotion plans for grocery retail stores. Bertsimas et al. (2016) use logistic regression to predict the outcomes of clinical trials testing combination chemotherapy regimens in order to design the best possible treatments against cancer.

Many advanced predictive models such as tree ensemble models and deep neural networks have proven to be popular and successful, far outside the realm of data science (Qi et al., 2019; Mignan and Broccardo, 2020; Delarue et al., 2020) and could be used inside of step (ii). In particular, tree ensembles rank among the best predictive models across a variety of datasets (Fernández-Delgado et al., 2014). The low entry barrier to learn and apply those models has accelerated their adoption in non-engineering fields such as biology and medical science. They however result in nonconvex discrete problems. Motivated by significant progresses in discrete optimization and by ever increasing computing power, global optimization perspectives are now taken on both steps (i) and (ii). As an example of step (i), Bertsimas and Dunn (2017) propose an integer programming approach to find optimal classification trees. This approach outperforms CART (classification and regression trees Breiman et al. (1984)), a leading decision tree method. These improvements are significant as optimal classification trees are used in many practically important applications, including the design of personalized medicine; see Bertsimas et al. (2019). As an example of step (ii), optimization of pre-trained tree ensembles has been used in the design of medicine (Mišić, 2020) and assortment planning (Chen and Mišić, 2021).

In this thesis, we introduce new results that contribute to the development of novel discrete optimization approaches to both step (i) the training of predictive models and step (ii) the optimization of problems with constraints/objectives described using pre-trained predictive models. We envision that these results would be utilized within the context of conjunction with the branch-and-cut framework (Conforti et al., 2014), which is currently the most common and most successful paradigm for the solution of discrete optimization problems. This paradigm is applicable to the solution of both mixed-integer linear and nonlinear programs (MILPs/MINLPs). It is implemented in state-of-the-art commercial solvers such as CPLEX (IBM, 2019) and Gurobi (Gurobi Optimization, LLC, 2021) for MILPs, and BARON (Sahinidis, 2017; Tawarmalani and Sahinidis, 2005) and ANTIGONE (Misener and Floudas, 2014) for MINLPs. The efficiency of branch-and-cut approaches rely on various crucial factors, among which are tight convex relaxations, effective cutting planes, high-quality heuristics, and proper branching strategies.

The first axis of this thesis is the study of *symbolic regression*. This is the problem of training a predictive model required to be a composition of given closed-form elementary functions to best fit given sample observations. A second axis of this thesis is the study of the structure of optimization models where pre-trained tree ensembles are used to describe constraints and/or objectives. This problem is known as *tree ensemble optimization*. We will show that this problem is deeply connected to multilinear optimization, *i.e.*, optimization models with objective/constraint functions that become linear when all but one of their variables are fixed.

The remainder of this thesis is organized as follows. In Chapter 2, we establish a reciprocity between tree ensemble optimization and multilinear optimization. We use this insight to derive new formulations for tree ensemble optimization problems and to obtain new convex hull results for multilinear polytopes. These results are obtained using a new convexification paradigm we introduce that is based on the novel concepts of *faciality*, *completability*, and *decomposition*. We then propose a generalization of decision trees, which we call *multilinear decision trees*, that allow complete reciprocity, even when multilinear functions are defined with respect to continuous variables. In addition to providing provably stronger formulations, we demonstrate numerically that these results help reduce solution times for multi-commodity transportation problems with piecewise-constant objective functions. In Chapter 3, we introduce strong formulations for piecewise polyhedral relaxations of multilinear optimization problems. These results can be used to relax tree ensemble optimization problems where the decision trees in the ensemble are multilinear decision trees, which we introduce in Chapter 2. The results we propose generalize those given in the literature in that they allow the partition over which these functions are defined to be nonregular. Numerical experiments with ALPINE, an open-source software for global optimization that relies on piecewise approximations of functions, further show that they significantly improve existing formulations. In particular, when enhanced with these results, ALPINE globally solves within one hour more instances (166 out of 220) than its default algorithm (64 out of 220) in computational experiments involving 220 hard multilinear and polynomial problem instances. In Chapter 4, we use discrete optimization techniques to develop strong MINLP formulations, cutting planes, and heuristics that result in improved algorithms for symbolic regression. Specifically, we improve MINLP formulations by removing redundant expression trees from the search space. We develop an MINLP-based heuristic that searches solutions by growing an expression tree. This heuristic is inspired by the fact that many physics formulas take simpler forms in specialized regimes. Focusing on discovering these simpler forms first, can then open the way to deriving the more general law. For example, one could think of deriving Kepler's third law first when $M \gg m$, leading to the expression $d = \sqrt[3]{c\tau^2 M}$. From this starting point, it is likely easier to discover the general law $d = \sqrt[3]{c\tau^2(M+m)}$. In computational experiments that are aimed at discovering physics formulas from sample data points, our method outperforms existing approaches based on MINLP or neural-networks both in finding a global solution given a restriction on the size of the expression tree and in finding good heuristic solutions for larger-sized expression trees. In Chapter 5, we conclude this thesis by providing a summary of its contributions and highlighting future avenues of research.

Chapter 2

A Reciprocity Between Tree Ensemble Optimization and Multilinear Optimization

The work described in this chapter was performed under the guidance of Prof. Jean-Philippe P. Richard and Prof. Mohit Tawarmalani. The research presented in this chapter was supported by grants 1727989 and 1917323 from the National Science Foundation Division of Civil, Mechanical and Manufacturing Innovation.

2.1 Preface

We establish a polynomial equivalence between tree ensemble optimization and optimization of multilinear functions over the Cartesian product of simplices. We use this insight to derive new formulations for tree ensemble optimization problems and to obtain new convex hull results for multilinear polytopes. A computational experiment on multi-commodity transportation problems with costs modeled using tree ensembles shows the practical advantage of our formulation relative to existing formulations of tree ensembles and other piecewise-linear modeling techniques. Motivated by the inability of tree ensembles to model multilinear functions of continuous variables, we then propose a generalization of decision-trees and provide empirical evidence of a better out-of-sample fit for a large collection of randomly perturbed nonlinear functions.

2.2 Introduction

Decision-trees and tree ensembles are commonly used predictive models for classification (when the output takes values in a finite discrete set) or for regression (when the output variable takes continuous values). They have been used in a variety of successful applications in medicine (Svetnik et al., 2003), marketing (Ferreira et al., 2016), engineering design (Deepa et al., 2010), drug development (Ma et al., 2015), and demand forecasting (Herrera et al., 2010). Tree ensembles rank among the best predictive model across a variety of datasets (Fernández-Delgado et al., 2014). On the other hand, multilinear optimization problems have been studied extensively in the mathematical programming literature (Tawarmalani and Sahinidis, 2002; Luedtke et al., 2012; Crama and Rodríguez-Heck, 2017; Del Pia and Khajavirad, 2018b,a; Xu et al., 2021; He and Tawarmalani, 2021). In particular, factorable programming problems are often modeled as multilinear programs with side constraints modeling univariate functions.

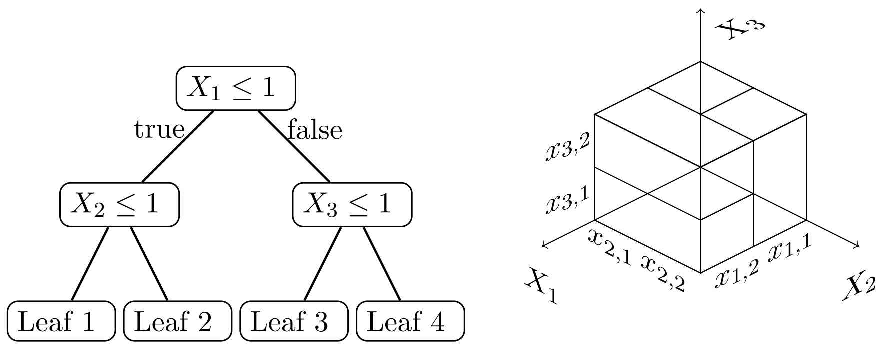
In this chapter, we study prescriptive models where the objective and/or constraints are modeled using tree ensembles. The idea of using mixed-integer programming (MIP) representations of pre-trained models is at the forefront of research (Anderson et al., 2020; Mišić, 2020). The special case that deals with optimizing an objective expressed as a pre-trained tree ensemble is referred to as the *tree ensemble optimization* (TEO) problem. Let T be the number of trees in the ensemble and let $f_t(X)$, for $t \in \{1, \dots, T\}$ be the t^{th} decision tree with input variables X. Then, TEO is formulated as

$$\max_{X} \quad \sum_{t=1}^{T} f_t(X). \tag{2.1}$$

Mišić (2020) was the first to formalize this problem, to establish its NP-hardness, and to propose MIP-based approaches for its solution. There is now a significant amount of work related to TEO; see our literature review in Section 2.3. TEO has been used in the design of medicine (Mišić, 2020) and assortment planning (Chen and Mišić, 2021). In the former, medical effects of candidates and in the latter consumer choices are modeled using random forests.

To the best of our knowledge, TEO and multilinear optimization have not been treated under a common umbrella before. Here, we show that understanding the connections between these problems is useful in extending state-of-the-art results for them. In particular, we develop strong formulations for functions described by pre-trained decision trees and tree ensembles using insights from multilinear optimization. These results follow from the following construction. Assume for simplicity that all variables are numerical as the ideas generalize trivially to the case of categorical variables. When the domain of input variables is a hyper-rectangle and all queries are of the form "`is` $X_v \leq c$?", decision trees model piecewise-constant functions defined over a hyper-rectangular partition $\mathcal{S}$ of the original domain. This piecewise-constant function can be modeled by introducing a binary variable for each partition element that indicates whether the input variables belong to the hyper-rectangle of $\mathcal{S}$ associated with this element. This hyper-rectangle is described using lower and upper bounds on its variables X. Projecting the partition elements along a variable X_v yields a discretization of the line-segment joining its upper and lower bound. For each interval in this discretization, we introduce an *interval indicator* binary variable which takes a value one when X_v belongs to that interval. We refer to the Cartesian product of intervals as a *partitioning atom*. Since each partition element is a union of partitioning atoms and the indicator for each atom is a product of interval indicators, the binary variable for each partition element is a sum of products of interval indicators, *i.e.* it is expressible as a multilinear function of interval indicators. Since the interval indicators for each variable belong to a simplex, $f_t(X)$ is a multilinear function of interval indicators over a Cartesian product of simplices. Therefore, problems involving decision trees are intimately related to multilinear optimization problems over Cartesian product of simplices; see Section 2.4 for more details. Figure 2.1 depicts the construction above, starting from a decision tree example defined over $\mathbb{R}^3$, displaying the corresponding hyper-rectangular partition, and producing the multilinear expression of an hyper-rectangle using binary elementary indicator variables.

Conversely, we will show in Section 2.4 that every multilinear function can be modeled as a tree ensemble of polynomial size. This establishes a reciprocity between the



(a) A decision-tree. (b) Hyper-rectangular partition.

Figure 2.1: A decision-tree (a) and its hyper-rectangular partition (b). The indicator of leaf 1 is $x_{1,1}x_{2,1}$.

two problems and allows us to develop various insights into these problems. In particular, a decision-tree view yields new polyhedral results on convexifying multilinear functions. Reciprocally, insights into convexification techniques for multilinear programs yield strengthened formulations for TEO and provide new ideas about how to use decision trees to develop better approximations of nonlinear functions.

The remainder of this chapter is organized as follows. In Section 2.3, we review related literature. In Section 2.4, we describe the problems formally and establish a precise reciprocity between them. In Section 2.5, we introduce a new convexification scheme for multilinear sets over the Cartesian product of simplices using notions of faciality, completability, and decomposition. Faciality (see Section 2.5.1) identifies multilinear functions with indicators of faces and develops polynomially-sized extended formulations of their convex hull and provides a closed-form projection to the space of original problem variables as well as conditions under which this projection is polynomially-sized. Our discussion on completability (see Section 2.5.2) expresses multilinear functions in a form that satisfies the faciality condition by introducing additional variables and discusses conditions when this completion is possible with a polynomial increase in variables. Decomposition techniques (see Section 2.5.3) focus on convexifying a multilinear set by independently convexifying suitable multilinear sets defined over subsets of the original variables. Together, these three tools provide powerful ways of developing convex hulls that generalize a variety of existing results in the literature. In Section 2.6, we apply the convexification ideas to develop improved formulations for TEO. For example, we give the first polynomial convex-hull formulations for a single decision tree with categorical or numerical variables. We then demonstrate on a multi-commodity flow application how these results can be used to create new stronger formulations for problems with piecewise-constant objectives. We conclude the chapter in Section 2.7.

The specific contributions of this chapter are as follows:

1. **Equivalence of two problems:** We show that TEO and multilinear optimization over the Cartesian product of simplices are equivalent. This connection reveals new ways of approaching convexification in multilinear optimization and tighter formulations for TEO.

2. **New convexification paradigm:** We introduce a new approach to simultaneously convexify multilinear sets over a Cartesian product of simplices. We derive new polynomially-sized convex hull descriptions for certain multilinear sets and show that many previously derived results follow readily from our approach.

3. **Multi-commodity transportation problem:** We solve, using various formulations, a transportation problem with piecewise-constant costs given via tree ensembles. Our formulation is among the smaller ideal formulations and, on large instances, performs better than piecewise-linear, disjunctive programming, and decision-tree based formulations.

4. **New approximation techniques for nonlinear functions:** We develop a generalized decision tree model to approximate nonlinear functions and their perturbations in Section 2.6.3. Computations confirm that the new tree structure provides better out-of-sample predictions.

2.3 Literature Review

TEO is first described and formalized in Mišić (2020), where the author shows that the problem is NP-Hard and gives both a linear and a nonlinear MIP formulation. The chapter also develops a Benders' decomposition algorithm for solving TEO. Chen and Mišić (2021) introduce an improved formulation for the case where all the independent input variables X are binary. Chen and Mišić (2022) applies these models and techniques to an assortment planning problem. In this context, each tree in the ensemble models the choice of a collection of customers given an offer set.

Mistry et al. (2021) introduces a variant of TEO to design a catalyst mixture, where the objective function is the sum of a tree ensemble function with a convex function of the continuous input variables X that are also used in the training of the tree ensemble. To formulate the problem, the authors introduce variables X in the model and relate them to the indicator variables used in Mišić (2020) through linear *linking constraints*. This formulation is then used inside of an optimization framework for the numerical solution of optimization problems defined through tree ensemble models in Thebelt et al. (2021). Biggs et al. (2017) study a variant of TEO where the branching decisions use general affine separators instead of threshold values for a single independent variable. In addition to providing an MIP model, the authors prove convergence results regarding the gap of the optimal value of the TEO model over a forest and that over a subset of trees.

Since TEO optimizes a piecewise-constant objective function, it can be viewed as a special case of piecewise-linear optimization. There is significant work on piecewise-linear functions in optimization; see Vielma et al. (2010); Vielma and Nemhauser (2011); Vielma (2015); Misener and Floudas (2012); Huchette and Vielma (2022); Baltean-Lugojan and Misener (2018), among others. We will develop formulations tighter than those in the TEO literature. These formulations also have better computational performance than disjunctive-programming-based formulations for a multi-commodity transportation problem where arc costs are piecewise-constant.

A function $f(x_1, \ldots, x_n)$ is *multilinear* if f is linear in each x_i, $i = 1, \ldots, n$ when x_j, for $j \neq i$, are fixed at some value. The problem of deriving strong and ideal formulations for the graphs/epigraphs of multilinear functions has been widely studied. When the domain of the function is $[0, 1]^n$, the convex hull is polyhedral and admits an exponentially-sized extended formulation; see Rikun (1997); Sherali (1997). This is because multilinear functions defined over hyper-rectangles are under-estimated by convex extensions of finite supports; see Tawarmalani and Sahinidis (2002) for a generalization. Crama (1993) identifies conditions under which the standard linearization of the multilinear function describes the envelope. Luedtke et al. (2012) studies McCormick's linearization of mutilinear functions and provide sufficient conditions under which it yields convex and concave envelopes. For bilinear functions whose variables belong to simplices, Bärmann et al. (2020) derive facet-defining inequalities of the bipartite boolean quadratic polytope with multi-choice constraints. Gupte et al. (2020) provide small-sized extended formulations for bilinear functions under certain conditions. The convex hull of permutation-invariant multilinear functions over a permutation-invariant box has been characterized; see Kim et al. (2021); Xu et al. (2021).

The simultaneous convex hull of multilinear functions over $[0,1]^n$ is also a polytope, which is typically referred to as the *multilinear polytope*. In particular, if $S = \{(\boldsymbol{x}, \boldsymbol{y}) \in \{0,1\}^{|V|} \times \mathbb{R}^{|E|} \mid y_e = \prod_{v \in e} x_v, \forall e \in E\}$, where $H = (V, E)$ is a hypergraph, then $\operatorname{conv}(S)$ is the multilinear polytope that describes the convex hull of $\prod_{v \in e} x_v, \forall e \in E$ over $[0,1]^n$. Standard linearization describes the convex hull of S when hypergraph H is Berge-acyclic; see Buchheim et al. (2019). Crama and Rodríguez-Heck (2017) introduces a family of valid inequalities called *2-link inequalities* for S that, when used in addition to standard linearization inequalities, are sufficient to describe its convex hull when H is laminar. Del Pia and Khajavirad (2018b) introduces a family of valid inequalities for S called *flower inequalities*. When used in conjunction with standard linearization, these inequalities provide a convex hull description of S when H is γ-acyclic. Finally, Del Pia and Khajavirad (2018a) show that, in some situations, the convex hull of a multilinear set S can be constructed by convexifying smaller sets separately. We borrow an idea from Schrijver (1983) to derive a more general decomposition result that can also be used to produce formulations of 0–1 multilinear sets with constant treewidth given in Bienstock and Munoz (2018).

2.4 Sources of Reciprocity Between the Two Problems

In this section, we establish an equivalence between TEO and multilinear optimization over the Cartesian product of simplices. We use bold lowercase letters to denote vectors. For positive integers a and b, we define $[a] := \{1, 2, \ldots, a\}$ and $[a..b] := \{a, a+1, \ldots, b\}$. For a positive integer K, we use $\Delta^K := \{\boldsymbol{x} \in \mathbb{R}_+^K \mid \sum_{j \in [K]} x_j = 1\}$ to denote the simplex having as vertices the K principal vectors of $\mathbb{R}^K$. We refer to the collection of these principal vectors as $\Delta_{0,1}^K$, *i.e.*, $\Delta_{0,1}^K = \Delta^K \cap \{0,1\}^K$.

2.4.1 TEO as a Multilinear Problem

Consider a tree ensemble model $f(X)$ with T binary decision trees, where, for each $t \in T$, $f_t(X)$ is a function of n independent variables $X = (X_1, \ldots, X_n)$. We derive a multilinear formulation for the associated TEO problem (2.1). We remark that the nonlinear formulation in Mišić (2020) is not multilinear in that variables belonging to the same simplex are multiplied with one another.

Let $\mathcal{N}$ and $\mathcal{C}$ denote the indices of numerical and categorical variables, respectively, and assume that $\mathcal{N} \cup \mathcal{C} = [n]$. The set of leaves (or leaf nodes) of tree t is denoted as $\textbf{leaves}(t)$ and the set of splits (or split/non-leaf nodes) of tree t as $\textbf{splits}(t)$. Each split s is assumed to involve a query that contains one independent variable, referred to as $\mathbf{v}(s)$. For $v \in \mathcal{N}$, the format of a query is "$X_v \leq a$?" where a is a scalar. We refer to this scalar a as the *split value* for independent variable X_v. For $v \in \mathcal{C}$, the format of a query is "$X_v \in A$?" where A is a subset of categories associated with X_v. We denote the set A associated with the query of split s by $\mathbf{c}(s)$.

To streamline the presentation, we make two assumptions that are without loss of generality. First, categorical variables X_v are relabeled to have categories $[K_v]$ for some positive integer K_v. Second, we assume $|\mathcal{N}| = 0$ because numerical variables can be modeled using categorical ones. To do so, for $v \in \mathcal{N}$, let $-\infty < a_{v,1} < a_{v,2} < \cdots < a_{v,K_v-1} < \infty$ denote the split values for X_v. Then, we may replace X_v with a categorical variable $\bar{X}_v$ by defining $(a_{v,j-1}, a_{v,j}] \cap \mathbb{R}$ as category j for $j \in \{1, \ldots, K_v\}$, where $a_{v,0} := -\infty$ and $a_{v,K_v} := \infty$; see Lemma 1 for a formal argument. The split query $X_v \leq a_{v,j}$? is the same as $\bar{X}_v \in [j]$? The categories created in this way have a special structure that we exploit in Section 2.5.1 to develop more compact models.

Let $x_{v,j}$, for $v \in [n]$ and $j \in [K_v]$, be binary variables that indicate if input variable X_v is assigned category j in a solution, *i.e.*, $x_{v,j} = \mathbb{1}[X_v = j]$. Since each input variable X_v is assigned to exactly one category, variables $\{x_{v,j}\}_{j \in [K_v]}$ belong to $\Delta_{0,1}^{K_v}$. With each $\ell \in \textbf{leaves}(t)$ we associate a binary variable $y_{t,\ell}$. Given X, we define $y_{t,\ell}$ to be one if and only if ℓ is the unique leaf in tree t such that all the queries from ℓ to the root are satisfied by X. Each leaf ℓ can be described by $(J_{t,\ell}^1, \ldots, J_{t,\ell}^n)$, where $J_{t,\ell}^v$ is the set of categories of input variable X_v that satisfy queries on X_v along the path from the root to ℓ. More specifically, $J_{t,\ell}^v = [K_v] \cap \left(\bigcap_{s \in M_{\text{y}}:\mathbf{v}(s)=v} \mathbf{c}(s)\right) \cap \left(\bigcap_{s \in M_{\text{n}}:\mathbf{v}(s)=v} [K_v] \setminus \mathbf{c}(s)\right)$, where $M_{\text{y}} \subseteq \textbf{split}(t)$ is the set of splits answered "yes" and $M_{\text{n}} \subseteq \textbf{split}(t)$ is the set of splits answered “no.”. Then, $y_{t,\ell} = \prod_{v\in[n]} \sum_{j\in J^v_{t,\ell}} x_{v,j}$. We now can formulate (2.1) as

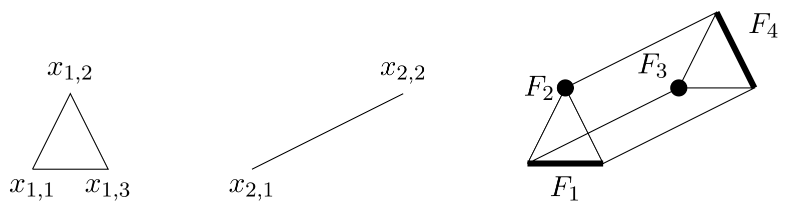
$$\max \left\{ \sum_{t\in[T]} \sum_{\ell\in\textbf{leaves}(t)} p_{t,\ell} y_{t,\ell} \;\middle|\; \begin{array}{l} y_{t,\ell} = \prod_{v\in[n]} \sum_{j\in J^v_{t,\ell}} x_{v,j}, \forall t\in[T],\ \forall \ell \in \textbf{leaves}(t), \\ \boldsymbol{x_v} \in \Delta^{K_v}_{0,1}, \forall v\in[n] \end{array} \right\} \tag{2.2}$$

where $p_{t,\ell}$ is the prediction value of leaf ℓ in tree t.

2.4.2 Multilinear Optimization Over the Cartesian Product of Simplices

We abstract (2.2) into a family of models that we call *multilinear optimization over the Cartesian products of simplices (MOCPS)*. These models have binary variables $\boldsymbol{x_v} = \{x_{v,j}\}_{j\in[K_v]} \in \Delta^{K_v}_{0,1}$ for all $v\in[n]$ so that $\boldsymbol{x_v}$ is a binarization of a variable with K_v possible values. We denote the extreme points of $P := \prod_{v\in[n]} \Delta^{K_v}$ by $D := \text{vert}(P)$. For each $(J^1,\dots,J^n)$ such that $J^v \subseteq [K_v]$, we associate the elementary multilinear function $\prod_{v\in[n]} \sum_{j\in J^v} x_{v,j}$. Any multilinear function of $\boldsymbol{x}$ is trivially an affine combination of elementary multilinear terms where $|J^v| = 1$ for all v. We associate each elementary multilinear function $g(\boldsymbol{x})$ with the face F of P it indicates, *i.e.*, $F = \{\boldsymbol{x}\in P \mid g(\boldsymbol{x}) = 1\}$.

Example 1. *Let $P_1 := \Delta^3$ and $P_2 := \Delta^2$. For $v = 1,2$, let $x_{v,j}$ be the j^{th} extreme point of P_v; see Figure 2.2a and Figure 2.2b. Then, the functions $\mathbb{1}_{F_1}(\boldsymbol{x}) = (x_{1,1}+x_{1,3})x_{2,1}$, $\mathbb{1}_{F_2}(\boldsymbol{x}) = x_{1,2}x_{2,1}$, $\mathbb{1}_{F_3}(\boldsymbol{x}) = x_{1,1}x_{2,2}$, and $\mathbb{1}_{F_4}(\boldsymbol{x}) = (x_{1,2}+x_{1,3})x_{2,2}$ indicate faces $F_1,\dots,F_4$ as shown in Figure 2.2c.*



(a) P_1 (b) P_2 (c) Faces $F_1,\dots,F_4$ of P.

Figure 2.2: Illustration of Example 1.

Given P and a set $\mathcal{F}$ of its faces, we define the multilinear set

$$\mathrm{ML}(D,\mathcal{F}) := \left\{(\boldsymbol{x},\boldsymbol{y}) \in D \times \mathbb{R}^{|\mathcal{F}|} \;\middle|\; y_F = \mathbb{1}_F(\boldsymbol{x}), \forall F \in \mathcal{F}\right\} \tag{2.3}$$

where $D = \mathrm{vert}(P)$ and $\mathbb{1}_F(\boldsymbol{x}) := \mathbb{1}[\boldsymbol{x} \in F]$. Given $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{F}|}$, we now define MOCPS as

$$\max\left\{\sum_{F\in\mathcal{F}} c_F y_F \;\middle|\; (\boldsymbol{x},\boldsymbol{y}) \in \mathrm{ML}(D,\mathcal{F})\right\}. \tag{2.4}$$

TEO, as in (2.2), is a special case of MOCPS because, for any leaf ℓ in tree t, $y_{t,\ell}$ is an elementary multilinear function determined by $\{J^v_{t,\ell}\}_{v\in[n]}$. Furthermore, MOCPS does not require that each vertex of P belongs to some face of $\mathcal{F}$ while TEO requires that each vertex is contained in one of the leaves for each tree. Clearly, MOCPS also generalizes unconstrained 0–1 multilinear optimization problems $\max_{\boldsymbol{x}\in\{0,1\}^n} \sum_{k\in[m]} c_k \prod_{v\in S_k} x_v$, where $S_k \subseteq [n]$ for $k \in [m]$ by choosing Δ^{K_v} to be Δ^2.

2.4.3 MOCPS as TEO

In Section 2.4.2, we showed that TEO is a special case of MOCPS. In this section, we derive the reverse implication by introducing at most a polynomial number of terms, thereby deriving an equivalence between the two problems that is important for the later sections. We assume that:

Assumption 1. TEO: *Each input variable is used in some branching query. Changing the category of an input variable leads it to a different leaf for some setting of the remaining variables.*

MOCPS: *For all v, $2 \le K_v \le |\mathcal{F}| + 1$.*

By simplifying instances, we can always ensure that Assumption 1 is satisfied. In fact, if TEO does not satisfy Assumption 1, we can either discard an input variable or merge two of its categories into one, without altering the predictions of the ensemble. If $K_v = 1$ in MOCPS, the variable v can be dropped. Further, for each $1 \le i,j \le K_v$ construct an edge between i and j if there exists a face $F \in \mathcal{F}$ with $|J^v \cap \{x_{v,i}, x_{v,j}\}| = 1$. Then, if $|\mathcal{F}| < K_v(K_v-1)/2$ there is a pair (i,j) that is not connected. Combining these categories does not alter MOCPS. Since $K_v \ge 2$, it follows that $|\mathcal{F}| \ge K_v - 1$.

The complexity of TEO is measured by $L := \sum_{t\in[T]} |\mathbf{leaves}(t)|$, the total number of leaves in the tree ensemble. This is because the numbers of trees, input variables, categories for each input variable, are bounded from above by L. The complexity of MOCPS can be measured using n and $|\mathcal{F}|$, the numbers of simplices and faces, respectively. This is because the number of vertices for all simplices are polynomially bounded in n and $|\mathcal{F}|$. When we differentiate the number of input variables n in TEO from the number of simplices n in MOCPS, we will use n_T and n_M, respectively.

Theorem 1 establishes a precise constructive polynomial equivalence between the two problems.

Theorem 1. *Any instance of TEO with L leaves, can be reduced into an instance of MOCPS with at most L simplices and at most L faces. Any instance of MOCPS with n simplices and a set $\mathcal{F}$ of faces can be reduced into an instance of TEO with at most $|\mathcal{F}|(n+1)$ leaves.*

Proof. Consider an instance of TEO with L leaves and n_T input variables. The reduction given in Section 2.4.2 reduces it to an instance of MOCPS with $|\mathcal{F}|$ faces and n_M simplices. Since the reduction models each leaf as an elementary multilinear term, then $|\mathcal{F}| = L$. Since one simplex is created for each input variable, we compute that $n_M = n_T \leq L$ under Assumption 1.

Consider an instance of MOCPS with n simplices so that $P := \prod_{v\in[n]} \Delta^{K_v}$, with a set $\mathcal{F}$ of faces in P characterized by subsets $J_F^v \subseteq [K_v]$, and with cost vector $\boldsymbol{c} \in \mathbb{R}^{|\mathcal{F}|}$. The objective of MOCPS is $\sum_{F\in\mathcal{F}} c_F \mathbb{1}_F(\boldsymbol{x})$. We will construct a tree ensemble with $|\mathcal{F}|$ trees whose input variables $X = (X_1, \ldots, X_n)$ are categorical and X_v has K_v category values for all $v \in [n]$. For $v \in [n]$ and $j \in [K_v]$, we think of $x_{v,j}$ in MOCPS as indicating whether input variable X_v takes category value j in TEO. For each face $F \in \mathcal{F}$, we construct a decision tree t_F with up to $n+1$ leaves, one of which models $c_F \mathbb{1}_F(\boldsymbol{x})$ in MOCPS; see Algorithm 1:

Algorithm 1: Construction of a decision tree associated with face F

```
    Data: Positive integer $K_v$, $J_F^v \subseteq [K_v]$ for all $v \in [n]$, and $c_F \in \mathbb{R}$.
    Result: A decision tree $t$.
1   $t \leftarrow$ single node decision tree;
2   $\ell \leftarrow$ root of $t$;
3   for $v \in [n]$ do
4       if $J_F^v \subsetneq [K_v]$ then
5           Create left and right children of $\ell$ with branching query "$X_v \in J_F^v$?";
6           $\ell \leftarrow$ the left child of $\ell$;
7   Set the prediction value of $\ell$ to $c_F$;
8   Set the prediction values of all other leaves to 0;
```

It is clear that the decision tree created by Algorithm 1 generates, for a value X of the input variables, an output value identical to $c_F \mathbb{1}_F(\boldsymbol{x})$. Figure 2.3 illustrates the decision tree created by Algorithm 1 for the face F where $\mathbb{1}_F(\boldsymbol{x}) = (x_{11} + x_{13})x_{22}(x_{42} + x_{44})$.

Let $\mathcal{T}$ be the tree ensemble composed of trees $\{t_F\}_{F \in \mathcal{F}}$ produced by Algorithm 1 for each $F \in \mathcal{F}$. Solving TEO for $\mathcal{T}$ is equivalent to solving the given instance of MOCPS because multilinear terms corresponding to any leaf which is not the leftmost one can be dropped because they have zero coefficients. After dropping these variables,

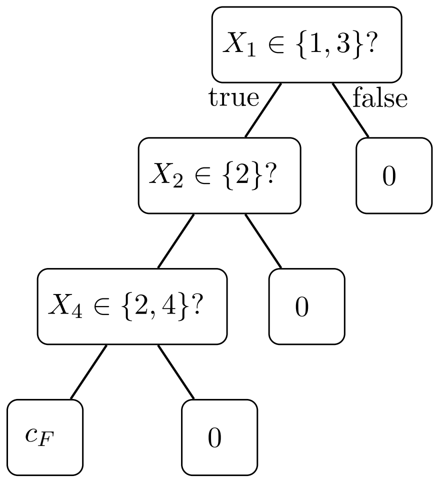


Figure 2.3: Decision tree associated with the face F where $\mathbb{1}_F(\boldsymbol{x}) = (x_{11}+x_{13})x_{22}(x_{42}+x_{44})$ with coefficient c_F.

the problem is identical to MOCPS. Because $\mathcal{T}$ has $|\mathcal{F}|$ trees and because each tree has at most $n+1$ leaves (as it branches at most n times according to Algorithm 1), we conclude that the total number of leaves of the trees in $\mathcal{T}$ is at most $|\mathcal{F}|(n+1)$. □

2.4.4 Supplements for Section 2.4

The following lemma explains why it is sufficient to consider a finite number of intervals for a numerical input variable.

Lemma 1. *Let $v \in [n]$ and $j \in [K_v]$. Assume that X' and X'' are such that (i) $X'_{v'} = X''_{v'}$ for $v' \neq v$ and (ii) $X'_v, X''_v \in (a_{v,j-1}, a_{v,j}] \cap \mathbb{R}$ where $a_{v,0} := -\infty$ and $a_{v,K_v} := \infty$. Then, $f(X') = f(X'')$.*

Proof. For $v' \neq v$, the answers to all queries on $X_{v'}$ over the trees for X' and X'' are identical because $X'_{v'} = X''_{v'}$. The answers to all queries on X_v over the trees for X' and X'' are identical because there are only $K_v - 1$ possible queries. For both X' and X'', the answer to query "$X_v \leq a_{v,p}$" is false when $1 \leq p \leq j-1$ and true when $j \leq p \leq K_v - 1$. Since the answers to all queries are identical for X' and X'' over all trees, each decision tree reaches the same leaf for X' and X''. Therefore, $f(X') = f(X'')$. □

Theorem 1 obviously implies that TEO is at least as hard as MOCPS. In the following sections we use Theorem 1 to transfer polyhedral results from one problem to another.

2.5 Facial Decomposition and its Convexification Properties

Given a set S and a *partition* $\mathcal{S} = \{S_1, \ldots, S_r\}$ of S, such that $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^r S_i = S$, we are interested in the simultaneous convex hull of indicators of S_i, $i = 1, \ldots, r$.

Definition 1. *The* partition indicator hull *of a set S, given its partition $\mathcal{S} = \{S_1, \ldots, S_r\}$, is the closed convex hull of $P(S, \mathcal{S})$, where $P(S, \mathcal{S}) := \{(\boldsymbol{x}, \boldsymbol{y}) \mid y_i = \mathbb{1}_{S_i}(\boldsymbol{x})$ for $i \in [r]$, $\boldsymbol{x} \in S\}$.*

Denote the closed convex hull of S_i as $\operatorname{cl}\operatorname{conv}(S_i)$ and the recession cone of $\operatorname{cl}\operatorname{conv}(S_i)$ as $O^+ \operatorname{cl}\operatorname{conv}(S_i)$. If $O^+ \operatorname{cl}\operatorname{conv}(S_i)$ does not vary with i, the partition indicator hull can be constructed using disjunctive programming techniques (Balas, 1985; Stubbs and Mehrotra, 1999; Ceria and Soares, 1999). In particular, let $X_i = P(S,\mathcal{S}) \cap \{\boldsymbol{y} \mid y_i = 1,\ y_j = 0 \text{ for } j \neq i\}$. Since, $P(S,\mathcal{S}) = \bigcup_{i=1}^{r} X_i$, it suffices to construct $\operatorname{cl}\operatorname{conv}(\bigcup_{i=1}^{r} \operatorname{cl}\operatorname{conv}(X_i))$. Since $X_i = \{(\boldsymbol{x},\boldsymbol{y}) \mid y_i = 1,\ y_j = 0 \text{ for } j \neq i,\ \boldsymbol{x} \in S_i\}$, then $O^{+}\operatorname{cl}\operatorname{conv}(X_i) = \{(\boldsymbol{x},\boldsymbol{y}) \mid \boldsymbol{x} \in O^{+}\operatorname{cl}\operatorname{conv}(S_i), y_j = 0 \text{ for } j \in [r]\}$. Since $O^{+}\operatorname{cl}\operatorname{conv}(S_i) = O^{+}\operatorname{cl}\operatorname{conv}(S_{i'})$ for all $i,\ i' \in [r]$, it follows that $O^{+}\operatorname{cl}\operatorname{conv}(X_i) = O^{+}\operatorname{cl}\operatorname{conv}(X_{i'})$. Then, Corollary 9.8.1 in Rockafellar (1970) shows that $\operatorname{conv}(\bigcup_{i=1}^{r} \operatorname{cl}\operatorname{conv}(X_i))$ is closed and has the same recession cone as $O^{+}\operatorname{cl}\operatorname{conv}(X_i)$. This convex hull can be computed using Theorem 9.8 in Rockafellar (1970).

Although the construction can be performed in general settings, we will consider the following special case. First, we will choose S to be the set of vertices of a Cartesian product of simplices. Although we restrict attention to vertices as in (2.3), we remark that the optimal value and the convex hull of the feasible region of MOCPS does not change if D is replaced with P; see Tawarmalani (2010). Second, these vertices are binary valued. Third, we require that $\operatorname{cl}\operatorname{conv}(S_i)$ has a succinct representation so that the disjunctive programming yields a tractable construction. Whenever possible, we will be interested in projecting the formulation back to the space of $(\boldsymbol{x},\boldsymbol{y})$ variables.

Given a polytope P with vertices in $\{0,1\}^n$, which we refer to as a *0–1 polytope*, we define specific partitions of interest that will be used to construct convex hulls.

Definition 2. *A* vertex decomposition *of a 0–1 polytope P is a partition of* $\operatorname{vert}(P)$ *into $\mathcal{V} = \{V_1,\dots,V_r\}$ and, for all i,* $\operatorname{conv}(V_i)$ *has a tractable $\mathcal{H}$-description.*

Definition 3. *A* facial decomposition *of a 0–1 polytope P is a vertex decomposition of* $\operatorname{vert}(P)$ *into $\mathcal{V} = \{V_1,\dots,V_r\}$ such that* $\operatorname{conv}(V_i)$ *is a face of P, whose defining inequality is available in closed-form. In other words, a facial decomposition may be described as $\mathcal{P} = \{P_1,\dots,P_r\}$ where each P_j is a hyperplane representation of the face* $\operatorname{conv}(V_j)$*.*

The key advantage of these partitions is that the resulting convex hull is integral. In Section 2.5.1, we derive convex hull representations for (2.3) when it corresponds to a partition indicator hull of $D := \prod_{v\in[n]} \Delta_{0,1}^{K_v}$. In this case, $\mathcal{F}$ is a facial decomposition of D since F is a face of $\operatorname{conv}(D)$ for all $F \in \mathcal{F}$. In Section 2.5.2, we discuss ways to construct a facial decomposition that refines a given set of faces. We show that there may not exist such a refined facial decomposition that is polynomially-sized in the number of faces provided. In Section 2.5.3, we identify conditions under which the convex hull of a given set can be constructed by deriving convex hulls of related sets with fewer variables and constraints. We then show that various results in the literature follow directly from our constructions. The proofs of all upcoming results can be found in 2.5.4.

2.5.1 Partition Indicator Hull of a Facial Decomposition

When the set $\mathcal{F}$ is a facial decomposition of D, we can express (2.3) as the following lifting of D:

$$\mathrm{ML}(D,\mathcal{F}) := \left\{ (\boldsymbol{x},\boldsymbol{y}) \in \prod_{v\in[n]} \Delta_{0,1}^{K_v} \times \mathbb{R}^{|\mathcal{F}|} \;\middle|\; y_F = \prod_{v\in[n]} \sum_{j\in J_F^v} x_{v,j}, \forall F \in \mathcal{F} \right\} \tag{2.5}$$

or, by defining $P_F := \{(\boldsymbol{x},\boldsymbol{y}) \mid (\boldsymbol{x},\boldsymbol{y}) \in \mathrm{ML}(D,\mathcal{F}),\ y_F = 1\}$ for all $F \in \mathcal{F}$, as $\mathrm{ML}(D,\mathcal{F}) = \bigvee_{F\in\mathcal{F}} P_F$. The convex hull of P_F can be described by intersecting P with $n+1$ linear constraints as follows:

$$\mathrm{conv}(P_F) = \left\{ (\boldsymbol{x},\boldsymbol{y}) \in \prod_{v\in[n]} \Delta^{K_v} \times \Delta^{|\mathcal{F}|} \;\middle|\; y_F = 1, \sum_{j\in J_F^v} x_{v,j} = 1, \forall v \in [n] \right\}. \tag{2.6}$$

Then, $\mathrm{conv}(\mathrm{ML}(D,\mathcal{F}))$ is obtained by applying disjunctive programming (see Balas, 1985, 1998). The resulting formulation has a polynomial number of variables and constraints.

Theorem 2. *It holds that* $\mathrm{conv}(ML(D,\mathcal{F}))$ *is the projection in the space of* $(\boldsymbol{x},\boldsymbol{y})$ *of* (2.7)*:*

$$\sum_{F\in\mathcal{F}:j\in J_F^v} z_{v,j,F} = x_{v,j}, \qquad \forall v \in [n],\ \forall j \in [K_v], \tag{2.7a}$$

$$\sum_{j\in J_F^v} z_{v,j,F} = y_F, \qquad \forall v \in [n],\ \forall F \in \mathcal{F}, \tag{2.7b}$$

$$\boldsymbol{x_v} \in \Delta^{K_v}, \qquad \forall v \in [n], \tag{2.7c}$$

$$\boldsymbol{y} \in \Delta^{|\mathcal{F}|}, \tag{2.7d}$$

$$z_{v,j,F} \geq 0, \qquad \forall v \in [n],\ \forall F \in \mathcal{F},\ \forall j \in J_F^v, \tag{2.7e}$$

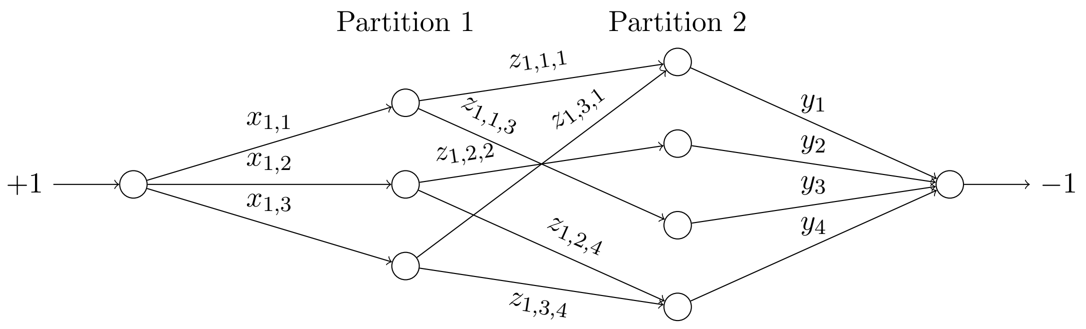


Figure 2.4: Transportation network corresponding to TP(1) for the example of Figure 2.2.

where (2.7c) *can be omitted, if desired.*

We next study the formulation of Theorem 2, which is unlike typical formulations constructed using disjunctive programming. In particular, it describes a network flow polytope with one source (resp. one destination) that has a supply (resp. a demand) of 1. The structure of this network flow problem is depicted in Figure 2.4. For each simplex, it models a transportation problem between nodes associated with the simplex variables $\boldsymbol{x}$ and nodes associated with indicator variables $\boldsymbol{y}$. Since $\boldsymbol{y} \in \Delta^{|\mathcal{F}|}$, which is a simplex, we can project the feasible region to $(\boldsymbol{x}, \boldsymbol{y})$ by checking whether it is feasible to transport, using $z_{v,j,F}$, a supply of $\boldsymbol{x_v}$ to meet demands of $\boldsymbol{y}$ independently for each $v \in [n]$. In particular, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \operatorname{conv}(\mathrm{ML}(D, \mathcal{F}))$ if and only if the transportation problems

$$\mathrm{TP}(v): \quad \sum_{F \in \mathcal{F}: j \in J_F^v} z_{v,j,F} = \bar{x}_{v,j}, \qquad \forall j \in [K_v], \tag{2.8a}$$

$$\sum_{j \in J_F^v} z_{v,j,F} = \bar{y}_F, \qquad \forall F \in \mathcal{F}, \tag{2.8b}$$

$$z_{v,j,F} \geq 0, \qquad \forall F \in \mathcal{F},\ \forall j \in J_F^v, \tag{2.8c}$$

are feasible for all $v \in [n]$. A consequence of Hoffman's circulation theorem (Hoffman (1976)) is that (2.8) is feasible if for each subset of supply nodes $\bar{N}$, the total demand for nodes that are connected to at least one supply in $\bar{N}$ does not exceed the total supply of $\bar{N}$. Thus, we obtain the following result.

Proposition 1. *Let N be any subset of $[n]$. Then,* conv$(ML(D,\mathcal{F}))$ *is described by:*

$$\sum_{F\in\mathcal{F}:J_F^v\subseteq J} y_F \le \sum_{j\in J} x_{v,j}, \qquad \forall v\in N,\ \forall\emptyset\neq J\subsetneq[K_v], \tag{2.9a}$$

$$\sum_{F\in H} y_F \le \sum_{j\in\bigcup_{F\in H}J_F^v} x_{v,j}, \qquad \forall v\in[n]\setminus N,\ \forall\emptyset\neq H\subsetneq\mathcal{F}, \tag{2.9b}$$

$$(\boldsymbol{x_1},\dots,\boldsymbol{x_n},\boldsymbol{y})\in\left(\prod_{v\in[n]}\Delta^{K_v}\right)\times\Delta^{|\mathcal{F}|}. \tag{2.9c}$$

There are exponentially many constraints in (2.9a) and (2.9b). The associated computational challenge is, however, mitigated by two factors. First, as described in 2.5.5, we can construct a polynomial time separation algorithm that uses (2.7) to generate cuts in the space of original problem variables. Second, some of the inequalities (2.9a) or (2.9b) are redundant. Although Theorem 1 in Kis and Horváth (2021) can be utilized to show this fact, the main idea in Proposition 2 is simply that (2.9a) (resp. (2.9b)) can, under certain conditions, be decomposed into two constraints of the same type.

Proposition 2. *Constraint (2.9a) for indices (v,J) is redundant if there is a non-trivial partition (J_1,J_2) of J (i.e., $J_1\neq\emptyset$ and $J_2\neq\emptyset$), where every $F\in\mathcal{F}$ such that $J_F^v\subseteq J$ satisfies either $J_F^v\subseteq J_1$ or $J_F^v\subseteq J_2$. Similarly, (2.9b) for indices (v,H) is redundant if there is a non-trivial partition (H_1,H_2) of H such that $\left(\cup_{F\in H_1}J_F^i\right)\cap\left(\cup_{F\in H_2}J_F^i\right)=\emptyset$.*

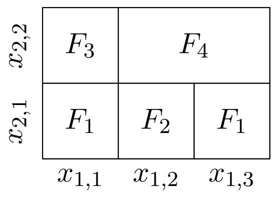
Using Proposition 1 and Proposition 2, we obtain the following formulation for conv(ML$(D,\mathcal{F})$) with the set of faces $\mathcal{F}$ presented in Figure 2.2:

$$y_3\le x_{1,1},\quad y_2\le x_{1,2},\quad y_1+y_3\le x_{1,1}+x_{1,3},\quad y_2+y_4\le x_{1,2}+x_{1,3}, \tag{2.10a}$$

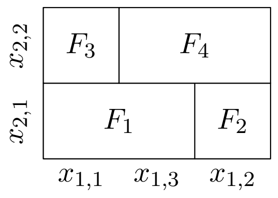
$$y_1+y_2\le x_{2,1},\quad y_3+y_4\le x_{2,2},\quad \boldsymbol{x_1}\in\Delta^3,\quad \boldsymbol{x_2}\in\Delta^2,\quad \boldsymbol{y}\in\Delta^4. \tag{2.10b}$$

We next describe sufficient conditions under which (2.9) reduces to a polynomially-sized formulation. These conditions encompass situations where the input variables of a decision tree are numerical, but also apply to certain situations where some variables are categorical.

Definition 4. *A collection $\mathcal{F}$ of subsets of vertices of* conv(D) *corresponding to faces has the* adjacency property *if for all $F\in\mathcal{F}$ defined using $(J_F^1,\dots,J_F^n)$, $J_F^v=[a_F^v..b_F^v]$*



(a) Original ordering.



(b) After switching $x_{1,2}$ and $x_{1,3}$.

Figure 2.5: Grid representations for Example 1 before and after reordering of variables.

for some $a_F^v, b_F^v \in [K_v]$.

If $K_v = 2$ for all $v \in [n]$, $\mathcal{F}$ always has the adjacency property because J_F^v is either $\{1\}$, $\{2\}$, or $\{1,2\}$, which are all sets of consecutive integers. We illustrate next that the adjacency property may hold after permuting the indices in $[K_v]$.

Example 2. *The set of faces in Example 1 does not satisfy the adjacency property. When we permute* $[K_1]$ *using* $\sigma_1 : [1,2,3] \to [1,3,2]$, *the same set of faces satisfies the adjacency property; see Figure 2.5 for an illustration.*

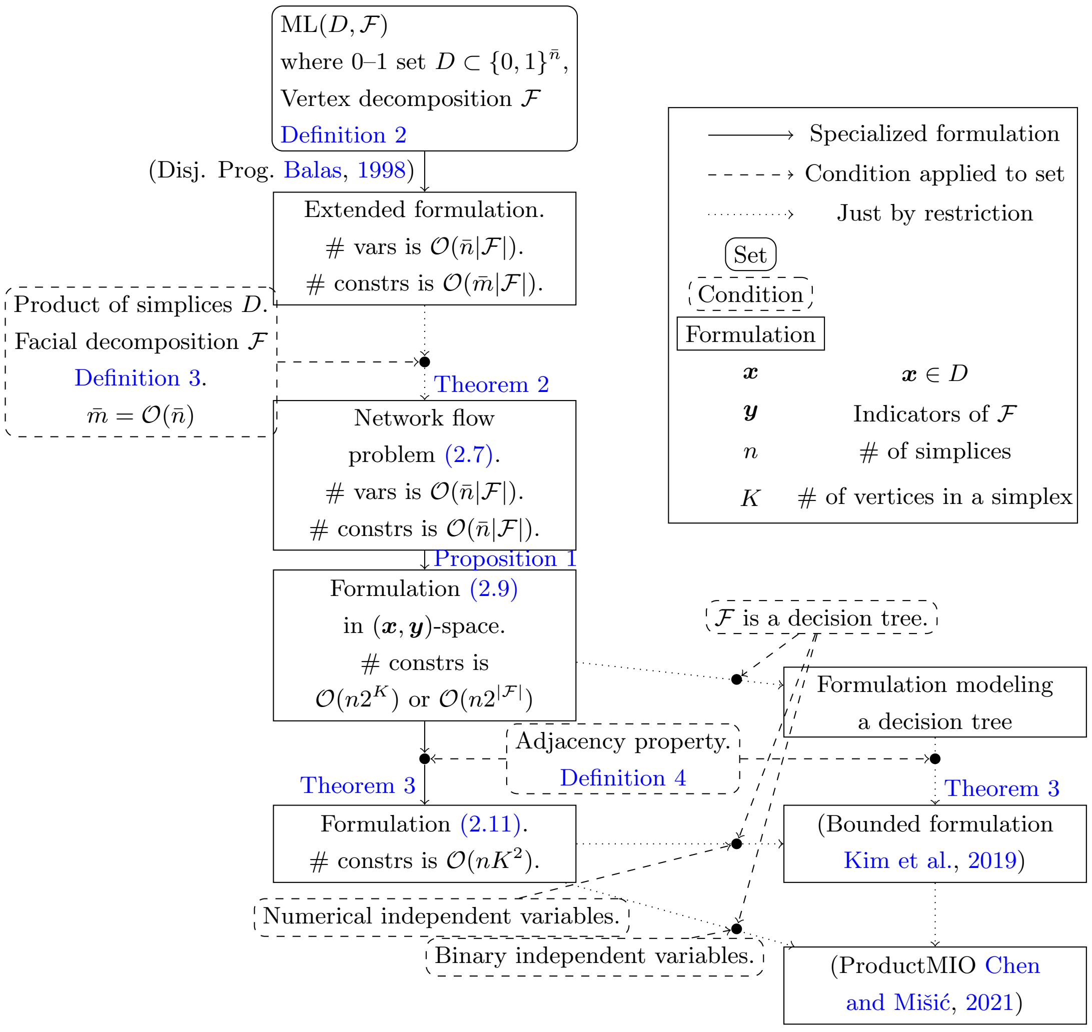
With the adjacency property, only polynomially many inequalities in (2.9) are non-redundant.

Theorem 3. *Suppose that (after suitable reordering)* $\mathcal{F}$ *has the adjacency property. Assume further that, for all* $v \in [n]$, $J_F^v = [a_F^v..b_F^v]$ *for some* $a_F^v, b_F^v \in [K_v]$. *Then, (2.11) describes* $\operatorname{conv}(ML(D,\mathcal{F}))$*:*

$$\sum_{F \in \mathcal{F} \,:\, J_F^v \subseteq [a..b]} y_\ell \le \sum_{j=a}^{b} x_{v,j}, \qquad \forall v \in [n],\ \forall a,b \in [K_v] : a \le b, \tag{2.11a}$$

$$(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}, \boldsymbol{y}) \in \left(\prod_{v \in [n]} \Delta^{K_v} \right) \times \Delta^{|\mathcal{F}|}. \tag{2.11b}$$

Example 3. *A description of the partition indicator hull of the facial decomposition presented in Example 1 is given in (2.10). Assume that the variables are first permuted as described in Example 2. Then, a simple inspection of (2.10) shows that the indices* (v, J) *for which the constraint (2.9a) is needed all satisfy the condition that* $J = [a..b]$ *for some* $a, b \in [K_v]$ *with* $a \le b$.



ML($D, \mathcal{F}$)
where 0–1 set $D \subset \{0,1\}^{\bar{n}}$,
Vertex decomposition $\mathcal{F}$
Definition 2
(Disj. Prog. Balas, 1998)
Extended formulation.
# vars is $\mathcal{O}(\bar{n}|\mathcal{F}|)$.
# constrs is $\mathcal{O}(\bar{m}|\mathcal{F}|)$.
Product of simplices D.
Facial decomposition $\mathcal{F}$
Definition 3.
$\bar{m} = \mathcal{O}(\bar{n})$
Theorem 2
Network flow
problem (2.7).
# vars is $\mathcal{O}(\bar{n}|\mathcal{F}|)$.
# constrs is $\mathcal{O}(\bar{n}|\mathcal{F}|)$.
Proposition 1
Formulation (2.9)
in $(\boldsymbol{x}, \boldsymbol{y})$-space.
# constrs is
$\mathcal{O}(n2^K)$ or $\mathcal{O}(n2^{|\mathcal{F}|})$
Specialized formulation
Condition applied to set
Just by restriction
Set
Condition
Formulation
$\boldsymbol{x}$
$\boldsymbol{x} \in D$
$\boldsymbol{y}$
Indicators of $\mathcal{F}$
n
# of simplices
K
# of vertices in a simplex
$\mathcal{F}$ is a decision tree.
Formulation modeling
a decision tree
Adjacency property.
Definition 4
Theorem 3
Theorem 3
Formulation (2.11).
# constrs is $\mathcal{O}(nK^2)$.
(Bounded formulation
Kim et al., 2019)
Numerical independent variables.
Binary independent variables.
(ProductMIO Chen
and Mišić, 2021)

Figure 2.6: Schematic illustration of relationships between results.

In Figure 2.6, we summarize the relations between the results obtained in this section and their connections to TEO that we will discuss in Section 2.6.1. To fix ideas, let $\bar{\boldsymbol{x}} \in \mathcal{H} := \{0,1\}^n$ and consider a vertex decomposition $\mathcal{F} = \{F_0, \ldots, F_n\}$ where $F_j = \big\{\bar{\boldsymbol{x}} \in \operatorname{conv}(\mathcal{H}) \mid \bar{x}_i = 1,\ \forall i \le j,\ \bar{x}_{j+1} = 0\big\}$ and where $\bar{x}_{n+1}$ is assumed to be zero. Clearly, $\operatorname{conv}\big(\mathrm{ML}(\mathcal{H}, \mathcal{F})\big)$ is the convex hull of $\big\{(\bar{\boldsymbol{x}}, \boldsymbol{y}) \in \mathcal{H} \times \mathbb{R}^{n+1} \mid y_j = (1-\bar{x}_{j+1}) \prod_{i=1}^{j} \bar{x}_i,\ \forall j \in [0..n]\big\}$. Each point in $\mathcal{H}$ belongs to the face F_j chosen so that $j+1$ is the smallest index with $\bar{x}_{j+1}$ equal to zero. Moreover since no vertex belongs to two of these faces, it follows that $\mathcal{F}$ is a facial decomposition. As is standard, binary variables x_j can be mapped to

Δ^2, and so to the setting of (2.4), by defining $x_{j,1} = \bar{x}_j$ and $x_{j,2} = 1 - \bar{x}_j$. Then, $\sum_{j \in J} x_{v,j}$ can be written as $0, x_{j,1}, x_{j,2}, 1$ when $J = \emptyset, \{1\}, \{2\}, \{1,2\}$, respectively. As stated earlier, all facial decompositions of $\mathcal{H}$ satisfy the adjacency property. This construction gives a simple way of constructing the convex hull of $\big\{(\bar{\boldsymbol{x}}, \boldsymbol{z}) \in \{0,1\}^n \times \mathbb{R}^n \mid z_j = \prod_{i=1}^{j} \bar{x}_i,\ \forall j \in [n]\big\}$ since $z_j = 1 - \sum_{j'=0}^{j-1} y_{j'}$.

As discussed above, Theorem 2 considers a special case where D, the extreme points of the Cartesian product of simplices, is endowed with a facial decomposition $\mathcal{F}$, and where the extended formulation of the partition indicator hull is a network flow problem. Then, using feasibility conditions, the formulation can be projected to the space of the original problem variables yielding (2.9). In the special case where the faces satisfy the adjacency property (Definition 4), this formulation is polynomially-sized. When modeling an instance of TEO with a single decision tree as $\text{ML}(D, \mathcal{F})$, the faces in $\mathcal{F}$ form a facial decomposition of $\text{conv}(D)$ since exactly one leaf becomes active for every choice of values for the input variables.

Remark 1. *The regions associated with the leaves of a decision tree form a facial decomposition.*

We will show in Section 2.6.1 that, when a decision tree has categorical variables, the network flow formulation of Theorem 2 gives the first polynomially-sized formulation for the convex hull of a single decision tree. For the case of numerical variables, the facial structure satisfies adjacency, and therefore, Theorem 3 provides a polynomially-sized formulation in the space of original problem variables. This yields the *bounded formulation* of Kim et al. (2019). The case where each input variable takes only two levels corresponds to the discussion about the hypercube $\mathcal{H}$ above. In this case, every facial decomposition satisfies the adjacency property. The bounded formulation thus yields the formulation of Chen and Mišić (2021).

As was alluded to, the results above allow us to model a decision tree. To do so, we choose D to be the vertices of the Cartesian product of simplices (describing the levels of each input variable) and $\mathcal{F}$ to be a facial decomposition of D (where each leaf node corresponds to a face of D.) We record in Remark 2 that the reverse mapping does not always hold. Here, we say that a facial decomposition $\mathcal{F}$ is *decision-tree-representable* if there exists a decision tree having a leaf for each $F \in \mathcal{F}$.

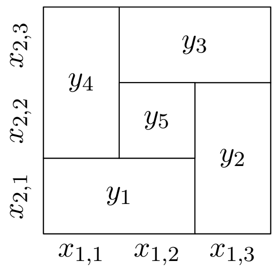


Figure 2.7: Facial decomposition that is not decision-tree representable. There is no way to partition $\{y_1, \ldots, y_5\}$ into two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ such that $\sum_{F\in\mathcal{F}_i} y_F$ corresponds to a face for $i = 1, 2$.

Remark 2. *Not every facial decomposition is decision-tree-representable.*

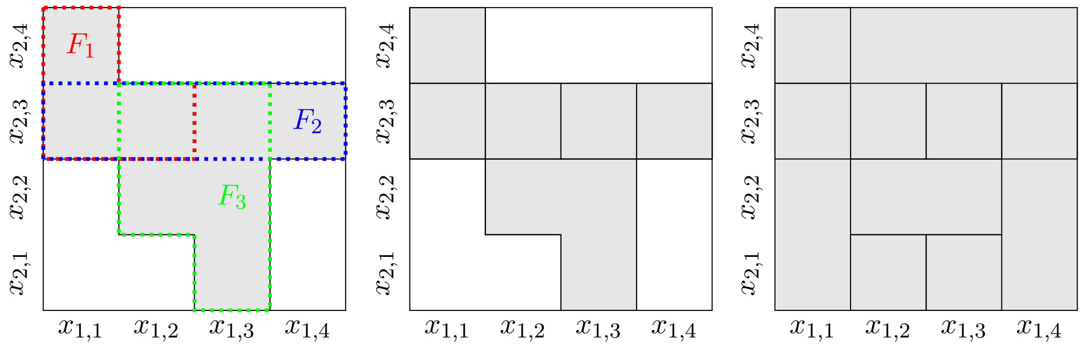
We provide an example to support Remark 2 in Figure 2.7. This contrasts with Theorem 1 that establishes that $\mathcal{F}$ can always be represented through a tree ensemble.

2.5.2 Completability: Construction of a Facial Decomposition

A given set of multilinear terms $\mathcal{F}$ in (2.4) may not describe a facial decomposition of the Cartesian product of simplices P as the faces may not cover $D := \text{vert}(P)$ and/or may overlap with one another. Given a set of faces $\mathcal{F}$ in P that meet certain technical requirements, we next construct a facial decomposition of P that can be used to derive an extended formulation for $\text{conv}(\text{ML}(D, \mathcal{F}))$. Such a facial decomposition could clearly be that which makes each vertex of P a face of $\mathcal{F}$, although this leads to exponentially-sized formulations. We show in Theorem 4 that, in general, there may not exist such a facial decomposition $\mathcal{F}'$ where the number of faces of $\mathcal{F}'$ is polynomial in n, $K^{\max}$, and $|\mathcal{F}|$ where $K^{\max} := \max_{v\in[n]} K_v$. However, we then identify special cases where such a polynomially-sized facial decomposition exists and provide algorithms to construct it. We now expand on the relationship between $\mathcal{F}$ and the desired facial decomposition, $\mathcal{F}'$.

Definition 5. *Let $\mathcal{F}$ be a set of integral polytopes in an integral polytope P. We say that $\mathcal{F}'$ is a* refinement *of $\mathcal{F}$ if (i) $\mathcal{F}'$ is a set of disjoint integral polytopes in P and (ii) every $F \in \mathcal{F}$ is the convex hull of a union of elements of $\mathcal{F}'$.*

Definition 6. *For an integral polytope P, we say that $\mathcal{F}'$ is a P*-completion *of a set $\mathcal{F}$ of integral polytopes in P if (i) $\mathcal{F}'$ is a refinement of $\mathcal{F}$ and (ii) $\mathcal{F}'$ is a facial decomposition*



(a) A set of integral polytopes $\mathcal{F}$ (b) A refinement of $\mathcal{F}$ (c) A P-completion of $\mathcal{F}$

Figure 2.8: Refinement and completion, using a grid representation.

of P.

Figure 2.8 shows a refinement (b) and a P-completion (c) where $P = \Delta^4 \times \Delta^4$, $\mathcal{F} = \{F_1, F_2, F_3\}$, $\mathbb{1}_{F_1}(\boldsymbol{x}) = (x_{1,1} + x_{1,2})x_{2,3} + x_{1,1}x_{2,4}$, $\mathbb{1}_{F_2}(\boldsymbol{x}) = x_{2,3}$, and $\mathbb{1}_{F_3}(\boldsymbol{x}) = (x_{1,2} + x_{1,3})(x_{2,2} + x_{2,3}) + x_{1,3}x_{2,1}$.

When $\mathcal{F}'$ is a refinement of $\mathcal{F}$, the variables y_F of $\mathrm{ML}(D, \mathcal{F})$ can be obtained as an affine transformation of the variables $y_{F'}$ of $\mathrm{ML}(D, \mathcal{F}')$. It follows, as we record in Proposition 3, that an extended formulation for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}))$ can be obtained from an LP description for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}'))$.

Proposition 3. *Let P be the Cartesian product of n simplices and let $D = \mathrm{vert}(P)$. Let $\mathcal{F}$ be a set of integral polytopes in P and $\mathcal{F}'$ be a refinement of $\mathcal{F}$. If there is an LP formulation T for $\mathrm{conv}(ML(D, \mathcal{F}'))$, then there is an LP formulation for $\mathrm{conv}(ML(D, \mathcal{F}))$, which adds $|\mathcal{F}|$ variables and $|\mathcal{F}|$ linear constraints to T.*

The proof of Proposition 3 is constructive. The size of the formulation obtained for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}))$ directly relates to that of $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}'))$. Since Proposition 1 presents an LP formulation for any facial decomposition $\mathcal{F}'$ that is polynomially-sized in n, $K^{\max}$, and $|\mathcal{F}'|$, the construction of Proposition 3 is small if we use a P-completion $\mathcal{F}'$ of $\mathcal{F}$ that has cardinality polynomial in n, $K^{\max}$, and $|\mathcal{F}|$, leading to

Definition 7. *A P-completion $\mathcal{F}'$ of a set of faces $\mathcal{F}$ in P is* polynomially-sized *if $|\mathcal{F}'|$ is polynomial in n, $K^{\max}$, and $|\mathcal{F}|$.*

Theorem 4 shows that polynomially-sized P-completion might not exist.

Theorem 4. *There is a family of Cartesian products P_n of n simplices, each with exactly two vertices, and a set $\mathcal{F}_n$ of faces in P_n such that there is no P_n-completion of $\mathcal{F}_n$ whose size is bounded above by a polynomial of n and $|\mathcal{F}_n|$.*

We next introduce cases in which a polynomially-sized LP formulation for $\text{conv}(\text{ML}(D, \mathcal{F}))$ can be derived by constructing a polynomially-sized P-completion of $\mathcal{F}$. We say that a set of integral polytopes $\mathcal{F}$ of P is *P-poly-completable* if there exists a polynomially-sized P-completion $\mathcal{F}'$ of $\mathcal{F}$.

Proposition 4. *Let P be the Cartesian product of n simplices and let $\mathcal{F}$ be a collection of vertices in P. Then, $\mathcal{F}$ is P-poly-completable and there is a completion whose size is at most $n|\mathcal{F}| + 1$.*

Proposition 5 shows that $\mathcal{F}$ is P-poly-completable as long as $|\mathcal{F}|$ is constant. The idea of the proof is as follows. First, a single face is P-poly-completable by Proposition 4 because it can be reduced to a vertex in the Cartesian product of simplices after suitable simplification so as to satisfy Assumption 1. Thus, we obtain $|\mathcal{F}|$ facial decompositions where each set has cardinality no more than $n+1$ and contains one of the faces in $\mathcal{F}$. To complete the proof, we introduce the notions of common P-completion and refinement. Given sets of faces $\mathcal{F}_1, \ldots, \mathcal{F}_T$ of P, $\mathcal{F}$ is a *common P-completion (resp. refinement) of $\mathcal{F}_1, \ldots, \mathcal{F}_T$* if $\mathcal{F}$ is a P-completion (resp. refinement) of $\mathcal{F}_t$ for all $t \in [T]$. We argue that, from the $|\mathcal{F}|$ facial decompositions of P, we can obtain a common P-completion (from their nonempty $|\mathcal{F}|$-wise intersections) that has no more than $(n+1)^{|\mathcal{F}|}$ elements.

Proposition 5. *A collection of a constant number of faces in a Cartesian product of simplices P is P-poly-completable.*

We next consider the case where $\mathcal{F}$ can be expressed by an arborescence, *i.e.*, a directed graph in which, for a node u called the *root* and any other node v, there is exactly one directed path from u to v. We say that $\mathcal{F}$ has the *arborescence property* if either $F_i \subseteq F_j$, $F_j \subseteq F_i$, or $F_i \cap F_j = \emptyset$ holds for all $F_i, F_j \in \mathcal{F}$. When $\mathcal{F}$ has the arborescence property, we can define *the graph of $\mathcal{F}$* as a directed graph G with $V(G) = \mathcal{F} \cup \{P\}$ and $E(G) = \{(v, w) \in V(G)^2 \mid w \subsetneq v$ and there is no $u \in V(G)$ such that $w \subsetneq$

$u \subsetneq v\}$. We say that *a face F_i (or its corresponding node) covers a face F_j (or its corresponding node)* if there is an arc (F_i, F_j) in the graph of $\mathcal{F}$.

Lemma 2. *Assume that $\mathcal{F}$ has the arborescence property. Then, the graph of $\mathcal{F}$ is an arborescence whose root is P.*

Consider now the graph G of a set of faces having the arborescence property. We define the set of outneighbors of node v as $\mathcal{N}_G^+(v) := \{w \in V(G) \mid (v,w) \in E(G)\}$. Let $u, w \in \mathcal{N}_G^+(v)$ be distinct. Then, $u \not\subseteq w$ and $w \not\subseteq u$. Therefore, $u \cap w = \emptyset$. Also, if $w \in \mathcal{N}_G^+(v)$ then $w \subsetneq v$. Therefore, for all $v \in V(G)$, $\bigcup_{w \in \mathcal{N}_G^+(v)} w \subseteq v$. If $\bigcup_{w \in \mathcal{N}_G^+(v)} w = v$ holds for all $v \in V(G)$, then the set of leaves of G, $\{v \in V(G) \mid \mathcal{N}_G^+(v) = \emptyset\}$, describes a facial decomposition. In this case, every node v is the union of all the leaves reachable from v. In Theorem 5, we give a sufficient condition for P-poly-completability of $\mathcal{F}$ when $\mathcal{F}$ has the arborescence property. The proof constructs an arborescence G' satisfying $\bigcup_{w \in \mathcal{N}_{G'}^+(v)} w = v$ for all $v \in V(G')$ by adding a polynomial number of nodes to the arborescence of $\mathcal{F}$. We will, with slight abuse of notation, say that a set of nodes in the arborescence of $\mathcal{F}$ is v-poly-completable if there is a v-completion of the faces associated with the set of nodes. We will also say that two nodes in the graph satisfy the arborescence property if one of the corresponding faces is contained in the other or if the faces are disjoint.

Theorem 5. *Let P be the Cartesian product of n simplices and $\mathcal{F}$ be a set of faces in P that has the arborescence property. If $\mathcal{N}_G^+(v)$ is v-poly-completable for all $v \in V(G)$ where G is the arborescence of $\mathcal{F}$, then there exists a P-completion $\mathcal{F}'$ of $\mathcal{F}$ where $|\mathcal{F}'|$ is polynomial in n, $K^{\max}$, and $|\mathcal{F}|$.*

Proposition 5 and Theorem 2 can be used to obtain polynomially-sized convex hull formulations for models where the prediction is obtained through boolean decision rules (rule sets in disjunctive normal form); see Su et al. (2016); Dash et al. (2018). This is because the trained model can be expressed as the indicator function of a constant number of faces. Finally, we mention that Angulo et al. (2015) studied a related problem regarding extended formulations for the convex hull of a subset of vertices of a polytope P. The authors show that when P is the unit hypercube $[0,1]^n$, polynomial formulations can be found and provide negative results for the case of general polytopes

P. Theorem 5 provides positive results for the case of polytopes P more general than the unit hypercube, and for the case where faces are not restricted to vertices.

Proof. We argue that for all $F_1, F_2 \in \mathcal{F}$, either $e(F_1) \subseteq e(F_2)$ or $e(F_2) \subseteq e(F_1)$ holds. By laminarity, for all $F_1, F_2 \in \mathcal{F}$, either $F_1 \subseteq F_2$, $F_2 \subseteq F_1$, or $F_1 \cap F_2 = \emptyset$ holds. If $F_1 \cap F_2 = \emptyset$, the argument holds by assumption. If $F_1 \subseteq F_2$, then $e(F_2) \subseteq e(F_1)$. For symmetry, if $F_2 \subseteq F_1$, then $e(F_1) \subseteq e(F_2)$. Therefore, the argument holds.

We next argue that there exists a permutation of $[n]$, $\sigma : [n] \mapsto [n]$, such that $e(F) = \{\sigma(1), \ldots, \sigma(|e(F)|)\}$ for all $F \in \mathcal{F}$. Since pairwise inclusion holds by the previous argument, we can order $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}$ so that $e(F_i) \subseteq e(F_j)$ for all $i < j$. Then, we create an order σ by assigning elements from $e(F_1), e(F_2) \setminus e(F_1), \ldots, e(F_{|\mathcal{F}|}) \setminus e(F_{|\mathcal{F}|-1})$ without duplicates. Without loss of generality, we assume that the current order of simplices satisfies $e(F) = \{1, \ldots, |e(F)|\}$ for all $F \in \mathcal{F}$.

We consider a decision tree T that splits P into faces. We only consider queries in the form of "$F \subseteq F_{v,j}$?" where $F_{v,j} := \{\boldsymbol{x} \in P \mid x_{v,j} = 1\}$. We order queries "$F \subseteq F_{v,j}$?" by increasing order of $v \in [n]$ and $j \in [K_v]$, that is, "$F \subseteq F_{1,1}$?", "$F \subseteq F_{1,2}$?", $\ldots$, "$F \subseteq F_{1,K_1}$?", "$F \subseteq F_{2,1}$?", $\ldots$, "$F \subseteq F_{n,K_n}$?". From a single node tree, we obtain T by recursively selecting a node whose corresponding region is not a vertex of P and splitting a node by choosing the first query that creates non-empty children from the order. For example, "$F \subseteq F_{1,2}$?" will not be chosen at a node if it previously answered "yes" to "$F \subseteq F_{1,1}$?" because the child followed answer "yes" to "$F \subseteq F_{1,2}$?" becomes empty ($F_{1,1} \cap F_{1,2} = \emptyset$). Therefore, every leaf corresponds to a vertex of P. Moreover, every node corresponds to a face whose indicator function is $x_{1,j_1} x_{2,j_2} \cdots x_{d,j_d}$ for some $d \in [n]$ and $j_1, \ldots, j_d \in [K_1] \times \cdots [K_d]$.

We next prune T. For all $F \in \mathcal{F}$, we mark a node of T whose corresponding region is F. Then, we recursively prune a pair of leaves if it belongs to the same parent and both are unmarked and remove the split information from the parent. Then, the remaining tree satisfies that if a leaf is unmarked, then its sibling node is marked. Therefore, the number of leaves is less than or equal to two times of the number of marked leaves. Since the number of marked leaves is bounded above by $|\mathcal{F}|$, the number of leaves in T is at most $2|\mathcal{F}|$. Let $\bar{\mathcal{F}}$ be the facial decomposition corresponding to the leaves of T. Then, $\bar{\mathcal{F}}$ implies $\mathcal{F}$ because there is a node whose corresponding region is F for every $F \in \mathcal{F}$ and every region of a node is implied by the leaves of its subtree. Therefore, the proof is done. □

2.5.3 Decomposability

We say a set S is *decomposable into sets S_1 and S_2* if $\mathrm{conv}(S) = \mathrm{conv}(S_1) \cap \mathrm{conv}(S_2)$. We study decomposability for $\mathrm{ML}(D, \mathcal{F})$ where $P_v = \Delta^{K_v}$ for all $v \in [n]$, $P = \prod_{v \in [n]} P_v$, $D = \mathrm{vert}(P)$, and $\mathcal{F}$ is a set of proper faces in P. Since P is a Cartesian product of sets, given a face F of P and $I \subseteq [n]$, we can write $F = F_I \times F_{\bar{I}}$, where F_I (resp. $F_{\bar{I}}$) is a face of $P_I := \prod_{v \in I} P_v$ (resp. $P_{\bar{I}} := \prod_{v \in \bar{I}} P_v$). Therefore, $\mathbb{1}_F(\boldsymbol{x}) = \mathbb{1}_{F_I}(\boldsymbol{x_I})\mathbb{1}_{F_{\bar{I}}}(\boldsymbol{x_{\bar{I}}})$. Since $F \in \mathcal{F}$ is a proper face of P, it is one of three types depending on whether $F_I = P_I$ and/or $F_{\bar{I}} = P_{\bar{I}}$. Using this classification, we define the *partition of $\mathcal{F}$ by I*, denoted as `partition`$(\mathcal{F}, I)$, as the triple $(\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2)$, where, a face $F \in \mathcal{F}^0$ (resp. $F \in \mathcal{F}^2$) satisfies $F_{\bar{I}} = P_{\bar{I}}$ (resp. $F_I = P_I$) and $\mathcal{F}^1 = \mathcal{F} \backslash (\mathcal{F}^0 \cup \mathcal{F}^2)$ consists of the remaining faces.

Example 4. *Let $P = (\Delta^2)^3 \times \Delta^3$ and $D = \mathrm{vert}(P)$. Consider*

$$T = \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in D \times \mathbb{R}^8 \;\middle|\; \begin{aligned} & y_1 = x_{11}x_{41}, \quad y_2 = x_{12}x_{21}(x_{42} + x_{43}), \\ & y_3 = x_{11}x_{42}, \quad y_4 = x_{11}x_{43}, \\ & y_5 = x_{12}x_{41}, \quad y_6 = x_{12}x_{22}(x_{42} + x_{43}) \\ & y_7 = x_{11}x_{31}, \quad y_8 = x_{11}x_{32} \end{aligned} \right\}.$$

For $j \in [8]$, let F_j be the face of P corresponding to y_j. Then, $T = ML(D, \mathcal{F})$ with $\mathcal{F} = \{F_1, \ldots, F_8\}$. Let $I = \{1, 2, 3\}$. For F_1, $\mathbb{1}_{(F_1)_I}(\boldsymbol{x}) = x_{11}$ and $\mathbb{1}_{(F_1)_{\bar{I}}}(\boldsymbol{x}) = x_{41}$. For F_7, $\mathbb{1}_{(F_7)_I}(\boldsymbol{x}) = x_{11}x_{31}$ and $\mathbb{1}_{(F_7)_{\bar{I}}}(\boldsymbol{x}) = 1$, i.e., $(F_7)_{\bar{I}} = P_{\bar{I}}$. Proceeding similarly with the other faces, we obtain that `partition`$(\mathcal{F}, I) = (\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2)$ *where $\mathcal{F}^0 = \{F_7, F_8\}$, $\mathcal{F}^1 = \{F_1, \ldots, F_6\}$, and $\mathcal{F}^2 = \emptyset$. In short, $\mathcal{F}^0$ (resp. $\mathcal{F}^2$) are the multilinear terms that do not involve variables $x_{4\cdot}$ (resp. $x_{i\cdot}$ for any $i \in I$).*

In the following result, we show that a multilinear set can sometimes be decomposed into two multilinear sets, by partitioning the variables into $(x_I, x_{\bar{I}})$. Then, we introduce variables for multilinear terms that depend on x_I and are involved in expressions involving both sets of variables. For a concrete illustration, in Example 4, if $I = \{1, 2, 3\}$, in order to separate the dependence of y_6 on $\boldsymbol{x_I} = \{x_1, x_2, x_3\}$ and $\boldsymbol{x_{\bar{I}}} = \{x_4\}$, we introduce a variable for $x_{12}x_{22}$. When it can be arranged that the introduced variables correspond to indicators of disjoint faces, possibly after refinement, Theorem 6 shows that the multilinear set is decomposable.

Theorem 6. *Consider a set of proper faces $\mathcal{F}$ of $P := \prod_{v\in[n]} P_v$ where $P_v := \Delta^{K_v}$ for all $v \in [n]$ and $I \subseteq [n]$. Express each $F \in \mathcal{F}$ as $F_I \times F_{\bar{I}}$ where F_I (resp. $F_{\bar{I}}$) is a face of P_I (resp. $P_{\bar{I}}$). Let $(\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2) = \texttt{partition}(\mathcal{F}, I)$, $\mathcal{F}^1_I := \bigcup_{F\in\mathcal{F}^1}\{F_I\}$, and assume that $\mathcal{F}'_I$ is a refinement of $\mathcal{F}^1_I$. For each $F \in \mathcal{F}^1_I$, let $C(F)$ be the set of faces in $\mathcal{F}'_I$ that refine F, i.e., $C(F) \subseteq \mathcal{F}'_I$ such that $\operatorname{conv}\big(\bigcup_{\hat{F}\in C(F)} \hat{F}\big) = F$. Introduce $\boldsymbol{z} \in \{0,1\}^{|\mathcal{F}'_I|}$ so that $\mathbf{1}^\intercal \boldsymbol{z} \le 1$ and, for every $F \in \mathcal{F}'_I$, let idx(F) be the index of F in $\mathcal{F}'_I$. Then*

$$ML(D,\mathcal{F}) = \operatorname{proj}_{\boldsymbol{x},\boldsymbol{y}}\left\{(\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}) \,\middle|\, (\boldsymbol{x_I},\boldsymbol{z},\boldsymbol{y^0}) \in ML(D_I,\mathcal{F}^0_I \cup \mathcal{F}'_I), (\boldsymbol{x_{\bar{I}}},\boldsymbol{z},\boldsymbol{y^1},\boldsymbol{y^2}) \in M\right\},$$

where $\mathcal{F}^0_I = \{F_I\}_{F\in\mathcal{F}^0}$, $\boldsymbol{y^0}, \boldsymbol{y^1}, \boldsymbol{y^2}$ correspond to indicators of $\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2$, respectively, and

$$M := \left\{ (\boldsymbol{x_{\bar{I}}},\boldsymbol{z},\boldsymbol{y}^1,\boldsymbol{y}^2) \,\middle|\, \begin{array}{ll} \boldsymbol{x_v} \in \Delta^{K_v}_{0,1}, & v \in \bar{I}, \\ \boldsymbol{z} \in \{0,1\}^{|\mathcal{F}'_I|}, \mathbf{1}^\intercal \boldsymbol{z} \le 1, & \\ y^1_F = \mathbb{1}_{F_{\bar{I}}}(\boldsymbol{x}) \displaystyle\sum_{\bar{F}\in C(F_I)} z_{idx(\bar{F})}, & F \in \mathcal{F}^1 \\ y^2_F = \mathbb{1}_{F_{\bar{I}}}(\boldsymbol{x}), & F \in \mathcal{F}^2 \end{array} \right\}.$$

Then, $\operatorname{conv}(ML(D,\mathcal{F})) = \operatorname*{proj}_{\boldsymbol{x},\boldsymbol{y}}\big\{(\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}) \mid (\boldsymbol{x_I},\boldsymbol{z}) \in \operatorname{conv}(ML(D_I,\mathcal{F}^0_I \cup \mathcal{F}'_I)), (\boldsymbol{x_{\bar{I}}},\boldsymbol{z},\boldsymbol{y}) \in \operatorname{conv}(M)\big\}$.

In words, Theorem 6 shows that it suffices to convexify $\mathrm{ML}(D_I, \mathcal{F}^0_I \cup \mathcal{F}'_I)$ and M separately in order to convexify $\mathrm{ML}(D,\mathcal{F})$. Further, since M can be expressed equivalently as $M = \mathrm{ML}\left(D_{\bar{I}} \times \Delta^{|\mathcal{F}'_I|+1}_{0,1}, \hat{\mathcal{F}}^2 \cup \hat{\mathcal{F}}^1\right)$, where $\hat{\mathcal{F}}^2 = \bigcup_{F\in\mathcal{F}^2} F_{\bar{I}} \times \Delta^{|\mathcal{F}'_I|+1}$, $\hat{\mathcal{F}}^1 = \bigcup_{F\in\mathcal{F}^1}\big\{F_{\bar{I}} \times \operatorname{conv}\big(\bigcup_{\bar{F}\in C(F)}\{e_{\mathrm{idx}(\bar{F})}\}\big)\big\}$, and $e_i \in \{0,1\}^{|\mathcal{F}'_I|+1}$ is the i^{th} principal vector, then M is a multilinear set of the form studied in Section 2.5.1.

Example 5. *Consider sets T and I defined in Example 4. Set $\mathcal{F}^1_I$ is the set of faces in P_I corresponding to $\{x_{11}, x_{12}, x_{12}x_{21}, x_{12}x_{22}\}$. The set of faces in P_I corresponding to $\{x_{11}, x_{12}x_{21}, x_{12}x_{22}\}$ is a P_I-completion of $\mathcal{F}^1_I$ that we use as $\mathcal{F}'_I$. This set $\mathcal{F}'_I$ is exactly that which would have been obtained using Theorem 5 since $\mathcal{F}^1_I$ satisfies the arborescence property and the internal nodes of the graph of $\mathcal{F}^1_I$ are completable by Proposition 4. Introducing variables $\boldsymbol{z} \in \Delta^3_{0,1}$, as specified in Theorem 6, to indicate $z_1 = x_{11}$, $z_2 =$*

$x_{12}x_{21}$, and $z_3 = x_{12}x_{22}$, we obtain that $\text{conv}(T)$ *is described as*

$$\text{proj}_{\boldsymbol{x},\boldsymbol{y}} \left\{ (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \;\middle|\; \begin{array}{l} \boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{x_3} \in \Delta^2, \boldsymbol{x_4} \in \Delta^3, \boldsymbol{y} \in \Delta^8, \boldsymbol{z} \in \mathbb{R}^3_+, \mathbf{1}^\intercal \boldsymbol{z} \le 1 \\ y_7 + y_8 \le x_{11}, z_2 + z_3 \le x_{12}, z_2 \le x_{21}, z_3 \le x_{22}, \\ y_7 \le x_{31}, y_8 \le x_{32}, z_1 = y_7 + y_8 \\ y_1 + y_5 \le x_{41}, y_3 \le x_{42}, y_4 \le x_{43}, y_2 + y_3 + y_4 + y_6 \le x_{42} + x_{43} \\ y_1 + y_3 + y_4 \le z_1, y_2 \le z_2, y_6 \le z_3, y_2 + y_5 + y_6 \le z_2 + z_3 \end{array} \right\}.$$

In obtaining this result, we use Theorem 3 twice to construct the convex hull over $(\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{x_3},y_7,y_8,\boldsymbol{z})$ and over $(\boldsymbol{x_4},\boldsymbol{z},y_1,\ldots,y_6)$. This is because (y_7,y_8,z_2,z_3) (resp. $(y_1,\ldots,y_6)$) are the indicators of a facial decomposition over $(\boldsymbol{x_1},\boldsymbol{x_2},\boldsymbol{x_3})$ (resp. over $(\boldsymbol{x_4},\boldsymbol{z})$) and both satisfy the adjacency property. The facial decomposition corresponding to (y_7,y_8,z_2,z_3) refines the faces corresponding to $(y_7,y_8,\boldsymbol{z})$ because $z_1 = y_7 + y_8$.

More generally, Theorem 6, when combined with the earlier results derived in Theorem 2 and Theorem 5, provides a powerful and systematic framework to construct convex hull descriptions for many multilinear sets. In particular, it is sufficiently general and versatile to provide alternate derivations and extensions of many results in the literature. As is commonly done, we may represent multilinear terms of a problem using a hypergraph where each vertex corresponds to a problem variable and where each hyperedge is incident to the variables nodes occurring in its corresponding monomial. As a first example, this framework can be used to derive polynomially-sized descriptions of convex hulls of certain multilinear sets whose underlying hypergraphs are laminar. This provides an alternate proof of Theorem 10 in Del Pia and Khajavirad (2018b). As a second example, this framework can be used to derive Theorem 1 in Del Pia and Khajavirad (2018a), a result that states that a convex hull description of the multilinear set associated with a hypergraph H can be obtained from convex hull descriptions of the multilinear sets associated with two smaller sub-hypergraphs H_1 and H_2 if their overlap is a hyperclique. As a third example, Theorem 3.5 in Bienstock and Munoz (2018) can be viewed as a special case of this framework. In particular, it allows for the construction of polynomially-sized formulations for the convex hull of multilinear sets whose structural sparsity is described by a tree-decomposition, when the tree-width is constant. A detailed discussion of these derivations is provided in 2.5.6. We however stress that the framework is in fact a strict generalization in that it allows for polynomially-sized formulations to be obtained for multilinear sets for which earlier results do not directly apply; see Example 6.

2.5.4 Supplements: Proofs of Statements in Section 2.5

This section includes proofs of theorems, propositions, lemmas introduced from Section 2.5.1 to Section 2.5.3.

Proof of Theorem 2

Proof. We apply disjunctive programming (Balas, 1985, 1998) to obtain a convex hull description of conv $\left(\bigvee_{F\in\mathcal{F}} P_F\right)$. First, recall that according to (2.6), $\operatorname{conv}(P_F)$ is described by the inequalities

$$\begin{aligned}
&\sum_{j\in J_F^v} x_{v,j} = 1, && \forall v\in[n], && (2.12a)\\
&x_{v,j} = 0, && \forall v\in[n],\ \forall j\notin J_F^v, && (2.12b)\\
&y_F = 1, && && (2.12c)\\
&y_{F'} = 0, && \forall F'\in\mathcal{F}\backslash\{F\}, && (2.12d)\\
&x_{v,j} \geq 0, && \forall v\in[n],\ \forall j\in[K_v] && (2.12e)
\end{aligned}$$

for each $F\in\mathcal{F}$. Next, we introduce a multiplier λ^F for each disjunct P_F and copies of the variables $(\boldsymbol{x},\boldsymbol{y})$ for each disjunct, which we denote by $(\boldsymbol{z_F},\dot{\boldsymbol{y}}^{\boldsymbol{F}})$, so as to obtain

$$\begin{aligned}
&\sum_{j\in J_F^v} z_{v,j,F} = \lambda^F, && \forall F\in\mathcal{F},\ \forall v\in[n], && (2.13a)\\
&z_{v,j,F} = 0, && \forall F\in\mathcal{F},\ \forall v\in[n],\ \forall j\notin J_F^v, && (2.13b)\\
&\dot{y}_F^F = \lambda^F, && \forall F\in\mathcal{F}, && (2.13c)\\
&\dot{y}_{F'}^F = 0, && \forall F\in\mathcal{F},\ \forall F'\in\mathcal{F}\backslash\{F\}, && (2.13d)\\
&z_{v,j,F} \geq 0, && \forall F\in\mathcal{F},\ \forall v\in[n],\ \forall j\in[K_v], && (2.13e)\\
&\lambda^F \geq 0, && \forall F\in\mathcal{F}, && (2.13f)\\
&\sum_{F\in\mathcal{F}} \lambda^F = 1, && && (2.13g)
\end{aligned}$$

$$x_{v,j} = \sum_{F \in \mathcal{F}} z_{v,j,F}, \qquad \forall v \in [n],\ \forall j \in [K_v], \tag{2.13h}$$

$$y_F = \sum_{F' \in \mathcal{F}} \dot{y}_F^{F'}, \qquad \forall F \in \mathcal{F}. \tag{2.13i}$$

Taken together, equalities (2.13c), (2.13d), and (2.13i) imply that $y_F = \dot{y}_F^F = \lambda^F$, which provides a way to eliminate variables $\dot{y}_F^{F'}$ and λ^F from the formulation. In particular, (2.13f) and (2.13g) become (2.7d). Similarly, (2.13a) becomes (2.7b). Using (2.13b) we eliminate zero variables from the formulation. In particular, (2.13h) becomes (2.7a) and (2.13e) implies (2.7e).

We conclude the proof by arguing that (2.7c) is redundant for each $v \in [n]$. First, observe that $\sum_{j \in [K_v]} x_{v,j} = 1$ in (2.7c), for each $v \in [n]$, is redundant as it can be obtained by summing (2.7a) for $j \in [K_v]$, subtracting (2.7b) for $F \in \mathcal{F}$, and subtracting $\sum_{F \in \mathcal{F}} y_F = 1$ in (2.7d). Second, $x_{v,j} \geq 0$ in (2.7c), for each $v \in [n]$ and $j \in [K_v]$, is also redundant as it can be obtained using (2.7a) and (2.7e). □

Proof of Proposition 1

Proof. Our proof of Proposition 1 relies on an ancillary result that we obtain first in Lemma 4 as an application of Hoffman's circulation theorem.

Lemma 3 (Hoffman (1976, Circulation Theorem))**.** *Let $G = (V, E)$ be a digraph. Let $\ell, u : E \mapsto \mathbb{R} \cup \{\pm\infty\}$ denote the lower and upper bound of flow on arc e. Assume that $\ell(e) \leq u(e)$ for every $e \in E$. Then, there exists a feasible flow $\phi : E \mapsto \mathbb{R}$ that satisfies (i) $\ell(e) \leq \phi(e) \leq u(e)$ for every $e \in E$ and (ii) $\sum_{j:(i,j)\in E} \phi(i,j) = \sum_{j:(j,i)\in E} \phi(i,j)$ for all $i \in V$ if and only if*

$$\sum_{(i,j)\in E: i \in X, j \notin X} \ell(i,j) \leq \sum_{(i,j)\in E: i \notin X, j \in X} u(i,j), \qquad \forall X \subseteq V. \tag{2.14}$$

Lemma 4. *Consider a bipartite graph G with $V(G) = U \cup W$ and $E(G) \subseteq U \times W$, supply $s_u \in \mathbb{R}_+$ for $u \in U$ and demand $d_w \in \mathbb{R}_+$ for $w \in W$. Assume that $\sum_{u \in U} s_u = \sum_{w \in W} d_w$. The associated transportation polytope is the set of solutions $\phi : E \mapsto \mathbb{R}_+$*

that satisfy the constraints

$$\sum_{(u,w)\in E:u=u'} \phi(u,w) = s_{u'}, \qquad \forall u' \in U, \tag{2.15a}$$

$$\sum_{(u,w)\in E:w=w'} \phi(u,w) = d_{w'}, \qquad \forall w' \in W. \tag{2.15b}$$

The transportation polytope is feasible if and only if

$$\sum_{w\in W'} d_w \le \sum_{u\in\bigcup_{w\in W'} U(w)} s_u, \qquad \forall W' \subseteq W, \tag{2.16}$$

where $U(w) := \{u \in U : (u,w) \in E(G)\}$ *for* $w \in W$*. Further* (2.16) *is equivalent to*

$$\sum_{w\in W:U(w)\subseteq U'} d_w \le \sum_{u\in U'} s_u, \qquad \forall U' \subseteq U. \tag{2.17}$$

Proof. We convert the solutions of the given transportation polytope into flows on a network N, for which Lemma 3 can be used. We define the vertex set $V(N) = U \cup W \cup \{s,t\}$ and the arc set $E(N) = E \cup E_s \cup E_t \cup \{(t,s)\}$ where $E_s = \{(s,u),\ \forall u \in U\}$ and $E_t = \{(w,t),\ \forall w \in W\}$. We set the lower and upper bounds on arcs of E to be 0 and ∞, respectively. For arcs (s,u) in E_s, these bounds are both set to s_u, while for arcs $(w,t) \in E_t$ they are both set to d_w. Finally the lower and upper bound on arc (t,s) are chosen to be $-\infty$ and ∞, respectively.

The transportation polytope and the set of feasible flows in N are equivalent because the flows on edges in $E_s \cup E_t \cup \{(t,s)\}$ are fixed. For $X \subseteq V(N)$, let $\delta^-(X) := \{(i,j) \in E(N) : i \notin X, j \in X\}$ and $\delta^+(X) := \{(i,j) \in E(N) : i \in X, j \notin X\}$. By Lemma 3, there exists a feasible flow in N if and only if

$$\sum_{(i,j)\in\delta^+(X)} \ell(i,j) \le \sum_{(i,j)\in\delta^-(X)} u(i,j), \qquad \forall X \subseteq V(N). \tag{2.18}$$

We next compute the left- and right-hand-side of (2.18) for each X in terms of s_u and d_w.

Condition (2.18) can be disregarded when its left-hand-side is equal to $-\infty$ and its right-hand-side is not, or when its right-hand-side is equal to ∞ and its left-hand-side is not. Further, observe that the left-hand-side (resp. right-and-side) is strictly less than ∞ (resp. strictly greater than $-\infty$) since $\ell(i,j) < \infty$ (resp. $u(i,j) > -\infty$) for all $(i,j) \in E(N)$. Since the left-hand-side becomes $-\infty$ when $\delta^-(X) \cap (E \cup \{(t,s)\}) \neq \emptyset$ and since the right-hand-side becomes ∞ when $\delta^+(X) \cap \{(t,s)\} \neq \emptyset$, we may assume from now on that (i) either $\{s,t\} \subseteq X$ or $X \cap \{s,t\} = \emptyset$, and (ii) $\bigcup_{w \in X \cap W} U(w) \subseteq X$. Under assumptions (i) and (ii), we can rewrite (2.18) in terms of s_u and d_w. There are two cases. If $\{s,t\} \subseteq X$, then

$$\sum_{w \in X \cap W} d_w \leq \sum_{u \in X \cap U} s_u. \tag{2.19}$$

If $X \cap \{s,t\} = \emptyset$, then

$$\sum_{u \in U \setminus X} s_u \leq \sum_{w \in W \setminus X} d_w. \tag{2.20}$$

Next, we argue that the relationships in (2.20) and some of the relationships in (2.19) are implied by a subset of conditions (2.19). To see this, observe first that, for any $X \subseteq V(N) \setminus \{s,t\}$ used in (2.20), the following holds because $\sum_{u \in U} s_u = \sum_{w \in W} d_w$, and sets U and W do not contain $\{s,t\}$:

$$\begin{aligned} \sum_{u \in U \setminus X} s_u \leq \sum_{w \in W \setminus X} d_w \quad &\Leftrightarrow \quad -\sum_{w \in W \setminus X} d_w \leq -\sum_{u \in U \setminus X} s_u \\ &\Leftrightarrow \quad \sum_{w \in W} d_w - \sum_{w \in W \setminus X} d_w \leq \sum_{u \in U} s_u - \sum_{u \in U \setminus X} s_u \Leftrightarrow \quad \sum_{w \in X \cap W} d_w \leq \sum_{u \in X \cap U} s_u \\ &\Leftrightarrow \quad \sum_{w \in (X \cup \{s,t\}) \cap W} d_w \leq \sum_{u \in (X \cup \{s,t\}) \cap U} s_u. \end{aligned}$$

From now, we can therefore assume that (iii) $\{s,t\} \subseteq X$ holds, which is stronger than and implies (i). For $X \subseteq V(N)$, let $U_X = X \cap U$ and $W_X = X \cap W$. We argue that (2.19) for $X \subseteq V(N)$ under (ii) and (iii) is implied by (2.19) for $X' \subseteq V(N)$ defined so that $U_{X'} = \bigcup_{w \in W_X} U(w)$ and $W_{X'} = W_X$. Set X' satisfies (ii) and (iii) by definition. Further, (2.19) for X' implies (2.19) for X because $W_X = W_{X'}$ by definition and because $U_{X'} \subseteq U_X$ (otherwise X would contradict assumption (ii).) In conclusion, there exists a feasible flow in N if and only if (2.19) holds for $X = W' \cup \bigcup_{w \in W'} U(w)$ for all $W' \subseteq W$. When expressed using s_u and d_w, this condition is (2.16).

We complete the proof by showing (2.16) and (2.17) are equivalent. First, we show that any inequality of (2.17) is implied by (2.16). Consider (2.17) for $U' \subseteq U$. Let $W' = \{w \in W : U(w) \subseteq U'\}$. It holds that $\bigcup_{w \in W'} U(w) \subseteq U'$ by definition. Hence, (2.17) for U' is implied by (2.16) for W'. Next, we show that any inequality of (2.16) is implied by (2.17). Consider (2.16) for $W' \subseteq W$. Let $U' = \bigcup_{w \in W'} U(w)$. It holds that $W' \subseteq \{w \in W : U(w) \subseteq U'\}$ by definition. Hence, (2.16) for W' is implied by (2.17) for U'. Therefore, (2.16) and (2.17) are equivalent. □

Recall the discussion before the introduction of (2.8) that point $(\boldsymbol{x}, \boldsymbol{y}) \in \prod_{v \in [n]} \Delta^{K_v} \times \Delta^{|\mathcal{F}|}$ is in conv(ML$(D, \mathcal{F})$) if and only if TP(v) given $(\boldsymbol{x_v}, \boldsymbol{y})$ is feasible for all $v \in [n]$. By Lemma 4, for $v \in [n]$, TP(v) is feasible if and only if $(\boldsymbol{x_v}, \boldsymbol{y})$ satisfies (2.16) or (2.17). Constraints (2.9a) and (2.9b) for $v \in [n]$ correspond to (2.17) and (2.16), respectively. Since it is enough to satisfy one of (2.9a) and (2.9b) for $v \in [n]$, system (2.9) becomes a formulation for conv(ML$(D, \mathcal{F})$) for any $N \subseteq [n]$. We exclude $J \in \{\emptyset, [K_v]\}$ and $H \in \{\emptyset, \mathcal{F}\}$ because the associated constraints reduce to either $0 \le 0$ or $1 \le 1$. □

Proof of Proposition 2

Proof. Consider first (2.9a) with indices (v, J). Assume that there exists a non-trivial partition (J_1, J_2) of J where every $F \in \mathcal{F}$ such that $J_F^v \subseteq J$ satisfies either $J_F^v \subseteq J_1$ or $J_F^v \subseteq J_2$. Then,

$$\sum_{F \in \mathcal{F}: J_F^v \subseteq J} y_F = \sum_{F \in \mathcal{F}: J_F^v \subseteq J_1} y_F + \sum_{F \in \mathcal{F}: J_F^v \subseteq J_2} y_F \le \sum_{j \in J_1} x_{v,j} + \sum_{j \in J_2} x_{v,j} = \sum_{j \in J} x_{v,j},$$

where the first equality holds by assumption, the second holds because $\sum_{F \in \mathcal{F}: J_F^v \subseteq J_k} y_F \le \sum_{j \in J_k} x_{v,j}$ is (2.9a) for indices (v, J_k) for $k = 1, 2$, and the last holds because $J_1 \cup J_2 = J$ and $J_1 \cap J_2 = \emptyset$. Therefore, (2.9a) for indices (v, J) is redundant since it can be obtained by summing (2.9a) for indices (v, J_k) for $k = 1, 2$.

Consider next (2.9b) with indices (v, H). Assume that there is a partition (H_1, H_2) of H where $(\cup_{F \in H_1} J_F^v) \cap (\cup_{F \in H_2} J_F^v) = \emptyset$. Then,

$$\sum_{F \in H} y_F = \sum_{F \in H_1} y_F + \sum_{F \in H_2} y_F \le \sum_{\cup_{F \in H_1} J_F^v} x_{v,j} + \sum_{\cup_{F \in H_2} J_F^v} x_{v,j} = \sum_{\cup_{F \in H} J_F^v} x_{v,j},$$

where the first equality holds by assumption, the second holds because $\sum_{F \in H_k} y_F \leq \sum_{\bigcup_{F \in H_k} J_F^v} x_{v,j}$ is (2.9a) for (v, H_k) for $k = 1, 2$, and the last holds because $\left(\bigcup_{F \in H_1} J_F^v\right) \cap \left(\bigcup_{F \in H_2} J_F^v\right) = \emptyset$ and $H = H_1 \cup H_2$. Therefore, (2.9b) for indices (v, H) is redundant since it can be obtained by summing (2.9b) for indices (v, H_k) for $k = 1, 2$. □

Proof of Theorem 3

Proof. By choosing $N = [n]$ in Proposition 1, it is sufficient to show that (2.9a) is implied by (2.11a). Consider a constraint (2.9a) with indices (v, J). If $J = [a..b]$ for some a and b, then it is clearly implied because there is an identical constraint in (2.11a).

Assume J is not of the form $[a..b]$. We say that $[a'..b']$ is a maximal consecutive integer subset of J if $[a'..b'] \subseteq J$ and $a' - 1, b' + 1 \notin J$. Then, there is a unique collection of maximal consecutive integer subsets of J, $\{[a_1..b_1], \cdots, [a_k..b_k]\}$, such that $J = [a_1..b_1] \cup \cdots \cup [a_k..b_k]$. Without loss of generality, assume that $a_1 < a_2 < \cdots < a_k$.

Pick any $F \in \mathcal{F}$ such that $J_F^v \subseteq J$. By the adjacency property, $J_F^v = [c..d]$ for some $c, d \in [K_v]$. Since $J_F^v \subseteq J$, $[c..d] \subseteq [a_r..b_r]$ for some $r \in [k]$. The left-hand-side of (2.11a) for (v, a_r, b_r) contains y_F. Also, the left-hand-side of (2.11a) for (v, a_p, b_p) for $p \neq r$ does not contain y_F. Therefore, y_F for all $F \in \mathcal{F}$ such that $J_F^v \subseteq J$ appears in exactly one of (2.11a) for $(v, a_1, b_1), \ldots, (v, a_k, b_k)$. It follows that the aggregation of the constraints of (2.11a) for $(v, a_1, b_1), \ldots, (v, a_k, b_k)$ results in

$$\sum_{F \in \mathcal{F} : J_F^v \subseteq J} y_F = \sum_{p=1}^{k} \left(\sum_{F \in \mathcal{F} : J_F^v \subseteq [a_p..b_p]} y_F \right) \leq \sum_{p=1}^{k} \left(\sum_{j \in [a_p..b_p]} x_{v,j} \right) = \sum_{j \in J} x_{v,j},$$

which is (2.9a) for (v, J). This completes the proof. □

Proof of Proposition 3

Proof. Since $\mathcal{F}'$ is a refinement of $\mathcal{F}$, there exists $A \in \mathbb{R}^{|\mathcal{F}| \times |\mathcal{F}'|}$ such that $\mathrm{ML}(D, \mathcal{F}) = \{(\boldsymbol{x}, A\boldsymbol{y'}) \mid (\boldsymbol{x}, \boldsymbol{y'}) \in \mathrm{ML}(D, \mathcal{F}')\}$. Since affine transformations and convex hull operators commute, it holds that $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F})) = \{(\boldsymbol{x}, \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y'}) \in \mathrm{conv}(\mathrm{ML}(D, \mathcal{F}')), \boldsymbol{y} = A\boldsymbol{y'}\}$. Therefore, $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}))$ can be formulated as an LP by adding $|F|$ variables $(\boldsymbol{y})$ and $|F|$ constraints $(\boldsymbol{y} = A\boldsymbol{y'})$ to the LP formulation for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}'))$. □

Proof of Theorem 4

Proof. Theorem 1.1 in Goos et al. (2018) proves that there exists a collection of simple graph $G_n = (V_n, E_n)$ for which every extended formulation of the independent set polytope of G_n has size (the total number of constraints) at least $\Omega(2^{n/\log n})$. Let $P_n = \prod_{v\in[n]} \Delta^2$ and let $D_n = \text{vert}(P_n)$. Let $\mathcal{F}_n = \{F_{i,j}\}_{(i,j)\in E_n}$ where $F_{i,j}$ is a face of P_n whose associated elementary multilinear term is $\mathbb{1}_{F_{i,j}}(\boldsymbol{x}) = x_{i,1}x_{j,1}$ for $(i,j) \in E_n$. The independent set polytope of G_n can be written as

$$I(G_n) = \text{proj}_{\boldsymbol{x}} \operatorname{conv}\left\{ (\boldsymbol{x}, \boldsymbol{y}) \in \text{ML}(D_n, \mathcal{F}_n) \,\middle|\, \sum_{(i,j)\in E_n} y_{i,j} = 0 \right\},$$

where $y_{i,j}$ is the indicator variable for $F_{i,j}$ for all $(i,j) \in E_n$. Since affine transformations commute with convexification, it holds that

$$I(G_n) = \text{proj}_{\boldsymbol{x}} \left\{ (\boldsymbol{x}, \boldsymbol{y}) \in \operatorname{conv}(\text{ML}(D_n, \mathcal{F}_n)) \,\middle|\, \sum_{(i,j)\in E_n} y_{i,j} = 0 \right\}.$$

Assume now by contradiction that there exists a P_n-completion $\mathcal{F}'_n$ of $\mathcal{F}_n$ of size bounded above by a polynomial of n. By Proposition 3, there exists a formulation for $\operatorname{conv}(\text{ML}(D_n, \mathcal{F}_n))$ whose size is polynomial in n, $|\mathcal{F}_n|$, and $|\mathcal{F}'_n|$. Since $|\mathcal{F}_n| \le \binom{n}{2}$ and $|\mathcal{F}'_n|$ is polynomial in n, the size of this formulation is polynomial in n. This contradicts the fact that the minimum size of extended formulations for the independent set polytope of G_n belongs to $\Omega(2^{n/\log n})$. $\square$

Proof of Proposition 4

Proof. The proof follows from Remark 1 by constructing a decision tree in which there is a leaf for each F in $\mathcal{F}$. We use induction to show that when $\mathcal{F}$ is a collection of vertices $v_1, \dots v_k$ of P, there exists a tree T with no more than $nk + 1$ leaves such each vertex v_j can be mapped to a leaf of T whose associated region is exactly the vertex. For the basis of induction, observe that, when $|\mathcal{F}| = 1$, Algorithm 1 can be used to produce a tree with the desired property that has no more than $n + 1$ leaves. For the inductive step, assume the result holds for collections of k vertices and consider a collection $\mathcal{F}$ such that

$|\mathcal{F}| = k+1$. For any vertex v in the collection $\mathcal{F}$, consider the collection $\tilde{\mathcal{F}}$ obtained by removing v from $\mathcal{F}$. By induction, there exists a tree $\tilde{T}$ with no more than $nk+1$ leaves such each vertex v_j can be mapped to a leaf of $\tilde{T}$ whose associated region is exactly the vertex. Next, identify the leaf ℓ' in $\tilde{T}$ whose associated region contains v. Applying lines 3-8 of Algorithm 1 by setting $\ell = \ell'$, we obtain a new tree T. This procedure does not add more than n leaves to $\tilde{T}$ showing that the number of leaves of T is bounded above by $n(k+1)+1$. Further, the analysis of Algorithm 1 performed in Theorem 1 also shows that one of these leaves has an associated region that is exactly vertex v. $\square$

Proof of Proposition 5

Proof. We first introduce the following ancillary result.

Lemma 5. *Let $\mathcal{F}_1, \ldots, \mathcal{F}_T$ be facial decompositions of P. Then, there is a common P-completion $\mathcal{F}$ of $\mathcal{F}_1, \ldots, \mathcal{F}_T$ such that $|\mathcal{F}| \leq \prod_{t=1}^{T} |\mathcal{F}_t|$.*

Proof. Define $\mathcal{F} := \bigcup_{(F_{1,j_1},\ldots,F_{T,j_T}) \in \mathcal{F}_1 \times \cdots \times \mathcal{F}_T} \{\bigcap_{t\in[T]} F_{t,j_t}\} \setminus \{\emptyset\}$. We claim that $\mathcal{F}$ is a common P-completion of $\mathcal{F}_1, \ldots, \mathcal{F}_T$. Collection $\mathcal{F}$ is a facial decomposition of P because (i) every element in $\mathcal{F}$ is a face in P as, in a polytope, the intersection of multiple faces is also a face, and (ii) every vertex $v \in \text{vert}(P)$ is contained in a unique face $F_i = F_{1,j_1} \cap \cdots \cap F_{T,j_T}$ in $\mathcal{F}$ where F_{t,j_t} is the unique face containing v for each $t \in [T]$ as $\mathcal{F}_t$ is a facial decomposition of P for all $t \in [T]$.

We next argue that $\mathcal{F}$ is a refinement of $\mathcal{F}_t$ for $t \in [T]$. Pick a face $F_{t^*,j^*} \in \mathcal{F}_{t^*}$. Let $\tilde{\mathcal{F}} = \bigcup_{v \in \text{vert}(F_{t^*,j^*})} \{F \in \mathcal{F} \mid v \in F\}$. It holds that $F_{t^*,j^*} \subseteq \text{conv}\left(\bigcup_{F\in\tilde{\mathcal{F}}} F\right)$ because there exists a face in $\mathcal{F}$ containing v for every vertex $v \in \text{vert}(P)$ by (ii). Also, $\text{conv}\left(\bigcup_{F\in\tilde{\mathcal{F}}} F\right) \subseteq F_{t^*,j^*}$ holds because every $F \in \tilde{\mathcal{F}}$ is obtained as $F_{1,j_1} \cap \cdots \cap F_{t^*,j^*} \cap \cdots \cap F_{T,j_T}$ for suitable faces $F_{t,j_t} \in \mathcal{F}_t$ for $t = 1, \ldots, t^*-1, t^*+1, \ldots, T$. Therefore, for any $t \in [n]$, any face in $\mathcal{F}_t$ can be expressed as the convex hull of a union of a subset of $\mathcal{F}$, *i.e.*, $\mathcal{F}$ is a refinement of $\mathcal{F}_t$ for all $t \in [T]$. It follows that $\mathcal{F}$ is a common P-completion of $\mathcal{F}_1, \ldots, \mathcal{F}_T$ with $|\mathcal{F}| \leq \prod_{t\in[T]} |\mathcal{F}_t|$, where the inequality holds by construction of $\mathcal{F}$. $\square$

Let $\mathcal{F} = \{F_1, \ldots, F_r\}$ be the given collection of a constant number of faces. Let n be the number of simplices. By Proposition 4, there exists a P-completion $\mathcal{F}_i$ of $\{F_i\}$ with $|\mathcal{F}_i| \leq n+1$ for all $i \in [r]$. By Lemma 5, there is a common P-completion $\mathcal{F}'$ of

$\mathcal{F}_1, \ldots, \mathcal{F}_r$ with $|\mathcal{F}'| \leq \prod_{i \in [r]} |\mathcal{F}_i| \leq (n+1)^r$. Then, $\mathcal{F}'$ is also a P-completion of $\mathcal{F}$ because $\mathcal{F} \subseteq \bigcup_{i \in [r]} \mathcal{F}_i$. Moreover, $|\mathcal{F}'|$ is polynomial in n since r is constant. Therefore, $\mathcal{F}$ is P-poly-completable. □

Proof of Lemma 2

Proof. Since each node is a subset of P, for each node $F \in \mathcal{F}$ we can find a sequence of sets ordered so that each set is covered by the previous set and the sequence begins at P and ends at F. This shows that P is connected to all the faces in $\mathcal{F}$. Now, we show that the path to each face F is unique. Instead, consider two paths $(R_1, \ldots, R_t)$ and $(S_1, \ldots, S_k)$ that are not identical. Assume further that $R_1 = S_1 = P$ and $R_t = S_k = F$ and without loss of generality that $t \geq k$. If the two paths are not identical, there is an $R' \in \{R_1, \ldots R_t\} \backslash \{S_1, \ldots, S_k\}$. Let R' be the highest indexed set that belongs to the first path but not the second. Then, it follows that the next set in the sequence, say R'' is contained in a set $S' \cap R'$, where we choose S' to be the highest indexed set in $\{S_1, \ldots, S_k\}$ that strictly contains R''. Since R'' is next to R' in one of the paths, $S' \not\subseteq R'$. Similarly, S' is next to R'' in $\{S_1, \ldots, S_k\}$, S' and, so, $R' \not\subseteq S'$. Moreover, since R' is not in the second path $R' \neq S'$. However, since $R'' \subseteq R' \cap S'$ and R'' is not empty, R' and S' violate the arborescence property. Therefore, the path from P to F must be unique. □

Proof of Theorem 5

Proof. Let G be the graph of $\mathcal{F}$. We construct an arborescence G' by adding nodes that correspond to faces of P so that, for all $v \in V(G')$, we have $\bigcup_{w \in \mathcal{N}^+_{G'}(v)} w = v$. Then, we show that the leaves of G', *i.e.*, $\mathcal{F}' = \{v \in V(G') \mid \mathcal{N}^+_{G'}(v) = \emptyset\}$, form a P-completion of $\mathcal{F}$ and that $|\mathcal{F}'|$ is polynomial in n, $K^{\max}$, and $|\mathcal{F}|$.

Let $G^1 = G$. We begin with $k = 1$. We let $I^1 = \{u \mid \mathcal{N}^+_{G^1}(u) \neq \emptyset\}$ be the set of internal nodes of G^1. Throughout the procedure, we ensure that (i) G^k is the graph of $V(G^k)$, (ii) the neighbors of any $u \in I^k$ remain unaltered so that it remains u-poly-completable, and (iii) if $u \in I^1 \setminus I^k$ then $\bigcup_{w \in \mathcal{N}^+_{G^k}(u)} w = u$. At each step, we pick $v \in I^k$. Since $\mathcal{N}^+_{G^k}(v)$ is v-poly-completable by our assumption, there exists a v-completion S_v of $\mathcal{N}^+_{G^k}(v)$ such that $\mathcal{N}^+_{G^k}(v) \subseteq S_v$ and $|S_v|$ is polynomial in n, $K^{\max}$, and $|\mathcal{N}^+_{G^k}(v)|$. We create G^{k+1} so that $V(G^{k+1}) = V(G^k) \cup S_v$ and $E(G^{k+1}) = E(G^k) \cup \{(v, v') \mid v' \in S_v\}$. We define $I^{k+1} = I^k \setminus \{v\}$.

We argue that (i) holds for G^{k+1}. To see this, we add nodes in S_v one by one. Let v' be a node newly added to the graph. We show that there cannot be a cover relation (w, u) so that, now $u \subseteq v' \subseteq w$. There are four cases to consider. If $w \cap v = \emptyset$, we cannot have $\emptyset \neq v' \subseteq v \cap w = \emptyset$. If $w \supsetneq v$, then $v \subseteq u$ which violates that $v' \subsetneq v$. If $w \subsetneq v$, there is a path in G^k from v to w. Let v'' be the node next to v on this path. Since $v' \neq v''$, we have $w \cap v' \subseteq v'' \cap v' = \emptyset$. Finally, we consider $v = w$. Then, $\emptyset \neq u = u \cap v' = \emptyset$. Therefore, v' does not have any outneighbors. Now, we show that v must have v' as its outneighbor in G^{k+1}. Assume instead that v' is the outneighbor of $w \neq v$. Since $v' \subseteq w \cap v$, $w \cap v$ cannot be empty. If $v \subsetneq w$, we would not connect w to v'. Therefore, $v' \subseteq w \subsetneq v$. However, then there is a node adjacent to v on the path from v to w in G^k, say v''. But, $v'' \cap v' = \emptyset$ since v' and v'' belong to a facial decomposition of v. Therefore, $w = v$.

We next argue that (ii) and (iii) hold for G^{k+1}. It follows from the above construction that for any $v' \in S_v \backslash \mathcal{N}^+_{G^k}(v)$, $\mathcal{N}^+_{G^{k+1}}(v') = \emptyset$. Therefore, the set of internal nodes of G^{k+1} is the same as that of G^k. For any $u \in I^{k+1}$, its neighbors in G^{k+1} are the same as those of u in G^k since $v \notin I^{k+1}$. Therefore, each node in $u \in I^{k+1}$ is still u-poly-completable, *i.e.*, (ii) holds. Moreover, for all $u \in I^1 \setminus I^{k+1}$, we have $\bigcup_{w \in \mathcal{N}^+_{G^{k+1}}(u)} w = u$ since the property remains true for nodes in $I^1 \backslash I^k$ as their neighbors were not altered and, for v the property was guaranteed by construction, *i.e.*, (iii) holds.

Now, we iterate with $k \leftarrow k + 1$. Since during each iteration the size of I^k reduces, the procedure converges in $t = |I^1|$ steps with a P-completion. This is because for any node $u \in G^t$ that is not a leaf, we have $\bigcup_{w \in \mathcal{N}^+_{G^t}(u)} w = u$. To verify that the leaves give a P-completion, observe that there is no path between the leaves of the arborescence. Therefore, the leaves are disjoint. Moreover, for any node in u that is not a leaf, there is an outneighbor of u that contains it. Therefore, recursively applying the idea, shows that there is a leaf that contains the node.

Now, we count the number of leaves in the arborescence at the end of the procedure. At each step, we add no more than $|S_v| - 1$ nodes, which is bounded above by a polynomial in n, $K^{\max}$, and $|\mathcal{F}|$, say $p(n, K^{\max}, |\mathcal{F}|)$. The maximum number of steps in the procedure is the number of internal nodes in G, which is bounded above by $|V(G)| = |\mathcal{F}|$. Therefore, $|\mathcal{F}'| \leq |\mathcal{F}| \times p(n, K^{\max}, |\mathcal{F}|)$ which is a polynomial in n, $K^{\max}$, and $|\mathcal{F}|$. $\square$

Proof of Theorem 6

Proof. The main idea of the proof is stated in Lemma 6. A similar decomposition principle has been used in several papers; see Schrijver (1983), for instance.

Lemma 6. *For $i = 1, 2$, let Q_i be an integral polytope in variables $(\boldsymbol{x_i}, \boldsymbol{y}) \in \mathbb{R}^{p_i} \times \mathbb{R}^{p_0}$. Assume that the projections of Q_1 and Q_2 in the space of $\boldsymbol{y}$ form a common simplex Δ. Then, polytope $S = \{(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y}) \mid (\boldsymbol{x_1}, \boldsymbol{y}) \in Q_1, (\boldsymbol{x_2}, \boldsymbol{y}) \in Q_2\}$ is integral.*

Proof. Given a point $(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y}) \in S$, we express it as a convex combination of points $(\boldsymbol{x_1^i}, \boldsymbol{x_2^i}, \boldsymbol{y^i})$ where $(\boldsymbol{x_1^i}, \boldsymbol{y^i})$ is an extreme point of Q_1 and $(\boldsymbol{x_2^i}, \boldsymbol{y^i})$ is an extreme point of Q_2. To do so, we consider the extreme points $\boldsymbol{y^k}$ of Δ. Then, we write $(\boldsymbol{x_1}, \boldsymbol{y}) = \sum_k \sum_{i \in I_k} \lambda_{i,k} (\boldsymbol{x_1^i}, \boldsymbol{y^k})$ and $(\boldsymbol{x_2}, \boldsymbol{y}) = \sum_k \sum_{j \in J_k} \gamma_{j,k} (\boldsymbol{x_2^j}, \boldsymbol{y^k})$. We can write $(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y}) = \sum_k \sum_{i \in I_k} \sum_{j \in J_k} \frac{\lambda_{i,k} \gamma_{j,k}}{\sum_{i \in I_k} \lambda_{i,k}} (\boldsymbol{x_1^i}, \boldsymbol{x_2^j}, \boldsymbol{y^k})$. To see that this works, observe that $\sum_{i \in I_k} \lambda_{i,k} = \sum_{j \in J_k} \gamma_{j,k}$, for each k, because $\{\boldsymbol{y^k}\}_k$ are the extreme points of a simplex. □

Let $\bar{M} := \mathrm{ML}(D_I, \mathcal{F}_I^0 \cup \mathcal{F}_I')$. Both $\mathrm{conv}(\bar{M})$ and $\mathrm{conv}(M)$ are integral polytopes. The projection in the space of $\boldsymbol{z}$ variables of $\mathrm{conv}(M)$ forms a simplex $\Delta^{\boldsymbol{z}} := \{\boldsymbol{z} \in \mathbb{R}_+^{|\mathcal{F}_I^1|} \mid \boldsymbol{1}^\intercal \boldsymbol{z} \le 1\}$. If $\bigcup_{F \in \mathcal{F}_I^1} F \subsetneq P_I$, then the projection in the space of $\boldsymbol{z}$ variables of $\mathrm{conv}(\bar{M})$ is also $\Delta^{\boldsymbol{z}}$. Then, Lemma 6 proves the result. Since $\bigcup_{F \in \mathcal{F}_I^1} F \subseteq P_I$ holds, it is sufficient to assume next that $\bigcup_{F \in \mathcal{F}_I^1} F = P_I$. The projection in the space of $\boldsymbol{z}$ variables of $\mathrm{conv}(\bar{M})$ is $\Delta^{\boldsymbol{z}} \cap \{\boldsymbol{z} \mid \boldsymbol{1}^\intercal \boldsymbol{z} = 1\}$. Let E be the hyperplane in the space of variables $(\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2})$ defined by the constraint $\boldsymbol{1}^\intercal \boldsymbol{z} = 1$ Then, it holds that $\mathrm{ML}(D, \mathcal{F}) = \{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \mid (\boldsymbol{x_I}, \boldsymbol{z}, \boldsymbol{y^0}) \in \bar{M}, (\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2}) \in M\} == \{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \mid (\boldsymbol{x_I}, \boldsymbol{z}, \boldsymbol{y^0}) \in \bar{M}, (\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2}) \in M \cap E\}$ since every point in the set satisfies $\boldsymbol{1}^\intercal \boldsymbol{z} = 1$. We complete the proof by arguing that

$$\begin{aligned}
&\mathrm{conv}(\mathrm{ML}(D, \mathcal{F})) \\
&= \mathrm{proj}_{\boldsymbol{x},\boldsymbol{y}} \big\{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \mid (\boldsymbol{x_I}, \boldsymbol{z}, \boldsymbol{y^0}) \in \mathrm{conv}(\bar{M}), (\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2}) \in \mathrm{conv}(M \cap E)\big\} \\
&= \mathrm{proj}_{\boldsymbol{x},\boldsymbol{y}} \big\{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \mid (\boldsymbol{x_I}, \boldsymbol{z}, \boldsymbol{y^0}) \in \mathrm{conv}(\bar{M}), (\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2}) \in \mathrm{conv}(M) \cap E\big\} \\
&= \mathrm{proj}_{\boldsymbol{x},\boldsymbol{y}} \big\{(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \mid (\boldsymbol{x_I}, \boldsymbol{z}, \boldsymbol{y^0}) \in \mathrm{conv}(\bar{M}), (\boldsymbol{x_{\bar{I}}}, \boldsymbol{z}, \boldsymbol{y^1}, \boldsymbol{y^2}) \in \mathrm{conv}(M)\big\}.
\end{aligned}$$

The first equality holds because of Lemma 6 and the fact that convex hull and projection operators commute. The second equality holds because E is a supporting hyperplane of

M and the convex hull of the intersection of a set and its supporting hyperplane equals to the intersection of the supporting hyperplane and the convex hull of a set. The third equality holds because $(\boldsymbol{x}_{\bar{\boldsymbol{I}}}, \boldsymbol{z}, \boldsymbol{y}^{\mathbf{1}}, \boldsymbol{y}^{\mathbf{2}}) \in E$ is implied by $(\boldsymbol{x}_{\boldsymbol{I}}, \boldsymbol{z}, \boldsymbol{y}^{\mathbf{0}}) \in \operatorname{conv}(\bar{M})$. □

2.5.5 Supplements: Polynomial Separation Algorithm

Proposition 6. *Let* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \prod_{v\in[n]} \Delta^{K_v} \times \Delta^{|\mathcal{F}|}$. *Verifying whether* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ *belongs to* $\operatorname{conv}(ML(D,\mathcal{F}))$ *can be done through the solution of the following separation problems*

$$
\begin{aligned}
z_v^* = \max \quad & \sum_{F\in\mathcal{F}} \bar{y}_F \mu_F - \sum_{j\in[K_v]} \bar{x}_{v,j}\lambda_{v,j} && && (2.21a)\\
s.t. \quad & \mu_F \le \lambda_{v,j}, && \forall F \in \mathcal{F},\ \forall j \in J_F^v, && (2.21b)\\
& \lambda_{v,j} \in [0,1], && \forall j \in [K_v], && (2.21c)\\
& \mu_F \in [0,1], && \forall F \in \mathcal{F}, && (2.21d)
\end{aligned}
$$

for $v \in [n]$. *In particular, if* $z_v^* = 0$ *for all* $v \in [n]$, *then* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \in \operatorname{conv}(ML(D,\mathcal{F}))$. *Otherwise,* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) \notin \operatorname{conv}(ML(D,\mathcal{F}))$ *and for any* $v \in [n]$ *such that* $z_v^* > 0$, *inequality*

$$
\sum_{F\in\mathcal{F}} \mu_F^* y_F \le \sum_{j\in[K_v]} \lambda_{v,j}^* x_{v,j}, \tag{2.22}
$$

where $(\boldsymbol{\lambda}_{\boldsymbol{v}}^*, \boldsymbol{\mu}^*)$ *is an optimal solution to* (2.21) *for* $v \in [n]$, *separates* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ *from* $\operatorname{conv}(ML(D,\mathcal{F}))$.

Proof. Choose $N = [n]$. According to Proposition 1, if $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ does not belong to $\operatorname{conv}(\mathrm{ML}(D,\mathcal{F}))$, then there exist $v \in [n]$ and $J \subseteq [K_v]$ such that

$$
\sum_{F\in\mathcal{F}: J_F^v \subseteq J} \bar{y}_F > \sum_{j\in J} \bar{x}_{v,j}. \tag{2.23}
$$

Since inequalities (2.9a) decompose by $v \in [n]$, it is sufficient to consider a separation problem for each $v \in [n]$:

$$
\begin{aligned}
\max \quad & \sum_{F\in\mathcal{F}} \bar{y}_F \mu_F - \sum_{j\in[K_v]} \bar{x}_{v,j}\lambda_{v,j} && && \\
\text{s.t.} \quad & \mu_F \le \lambda_{v,j}, && \forall F \in \mathcal{F},\ \forall j \in J_F^v, && (2.24a)
\end{aligned}
$$

$$\lambda_{v,j} \in \{0,1\}, \qquad \forall j \in [K_v], \tag{2.24b}$$

$$\mu_F \in \{0,1\}, \qquad \forall F \in \mathcal{F}. \tag{2.24c}$$

Constraint (2.24a) imposes that if y_F appears on the left-hand-side of (2.9a), then $x_{v,j}$ for all $j \in J_F^v$ appears on the right-hand-side. Constraints (2.24b) and (2.24c) impose that the coefficients of $x_{v,j}$ and y_F are either 0 or 1. The objective function represents the amount of the violation at $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. Therefore, given $v \in [n]$, problem (2.24) finds an inequality of the form (2.22) that $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ violates the most. Finally, binary conditions in (2.24) can be relaxed as the constraint matrix is totally unimodular. □

2.5.6 Supplements: Connecting our Framework to the Literature

In this section, we show that Theorem 6 can be applied to obtain existing results about decomposability or polynomially-sized convex hull formulations for certain multilinear sets over 0−1 variables given in the literature. We also present an example that shows that our results provide a strict generalization.

Let $S = \{(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in \{0,1\}^n, y_j = \prod_{i \in e_j} x_i, \forall j \in [m]\}$ where $e_j \subseteq [n]$ for $j \in [m]$. Set S is commonly represented using a hypergraph H where $V(H) = [n]$ and $E(H) = \bigcup_{j \in [m]} \{e_j\}$. For this reason, we refer to S as the multilinear set associated with hypergraph H. We denote by $H[V']$ the section hypergraph of H induced by a subset of nodes $V' \subseteq V(H)$.

The multilinear set of a laminar hypergraph

Theorem 10 in Del Pia and Khajavirad (2018b) provides a polynomially-sized formulation for the convex hull of the multilinear set S associated with a laminar hypergraph H. A hypergraph H is laminar if for every $e_i, e_j \in E(H)$, it holds that either $e_i \subseteq e_j$, $e_j \subseteq e_i$, or $e_i \cap e_j = \emptyset$. This result is a special case of Theorem 6. In fact, if a hypergraph H is laminar, $E(H)$ has the arborescence property. Set $V = \{\{v\}\}_{v \in V(H)} \cup E(H)$ has the arborescence property since H is laminar. Therefore, we can construct an arborescence G using Lemma 2 such that $V(G) = V$ and for every $u \in V(G)$, it holds that $z_u = \prod_{w \in V(G):(u,w) \in E(G)} z_w$ where, for each $u \in \{\{v\}\}_{v \in V(H)}$, z_u is x_u if $u \in V(H)$ and

z_u is y_u if $u \in E(H)$. This arborescence-structured dependency allows S to be decomposed based on any arc $(u, w) \in E(G)$ into S_1 and S_2 where S_1 is the multilinear set containing z_w together with the multilinear terms associated with the descendants of w, and where S_2 is the set containing the remaining multilinear terms. Theorem 6 can then be applied without further refinement as S_2 depends on the single multilinear term z_w in S_1. Since both S_1 and S_2 can be expressed as the multilinear sets of laminar hypergraphs, the procedure can be applied recursively. At termination, we have expressed the convex hull of S using a polynomially-sized collection of convex hulls of multilinear sets with a single multilinear term, which can be easily convexified using the standard linearization.

Next, we restate Theorem 10 in Del Pia and Khajavirad (2018b) in Corollary 1. We then provide a formal constructive proof that is a direct application of Theorem 6.

Corollary 1 (Theorem 10 in Del Pia and Khajavirad (2018b)). *Let H be a laminar hypergraph. There exists a polynomially-sized formulation for the convex hull of the multilinear set S associated with H.*

Proof. Consider the multilinear set S of a hypergraph H and assume that H is laminar. Without loss of generality, we assume that H is connected as S can be convexified by treating each component separately otherwise. We claim that $[n] \in E(H)$ if H is laminar and connected. Assume by contradiction $[n] \notin E(H)$. Pick $e_k \in E(H)$ such that $|e_k| = \max_{j\in[m]} |e_j|$. By the assumption, $e_k \subsetneq [n]$. Pick $i \in [n] \setminus e_k$. Since H is laminar and there is no e_j such that $e_k \subsetneq e_j$, there is no $e_\ell \in E(H)$ that contains i and any node in e_k. Moreover, this holds for all $j \in [n] \setminus e_k$. Therefore, node i and the nodes in e_k are disconnected, which is a contradiction to the assumption that H is connected. This establishes that $[n] \in E(H)$. We can now construct, using Lemma 2, an arborescence G from $E(H)$ where the root is $[n]$. Select a leaf e_k from G and decompose S into S_1 and S_2 using Theorem 6 with $I = e_k$. Consider $\mathrm{ML}(D, \mathcal{F})$ corresponding to S. Since e_k is a leaf, for all $e_j \in E(H)$, it holds either $e_k \subseteq e_j$ or $e_k \cap e_j = \emptyset$. Let $\bar{F}$ be the face corresponding to e_k and $\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2 = \texttt{partition}(\mathcal{F}, I)$. Then, $\mathcal{F}^0 = \{\bar{F}\}$, $\mathcal{F}^1 \cup \mathcal{F}^2 = \mathcal{F} \setminus \{\bar{F}\}$, and $\mathcal{F}^1_I = \{\bar{F}\}$. Since $\mathcal{F}^1_I$ is a disjoint set, we can apply Theorem 6 to decompose S into S_1 and S_2 where S_1 is the multilinear set of a hypergraph H_1 with $V(H_1) = e_k$ and $E(H_1) = \{e_k\}$ and S_2 is the multilinear set of a hypergraph H_2 with $V(H_2) = V(H) \setminus e_k \cup \{e_k\}$ and $E(H_2) = \{e \setminus e_k \cup \{e_k\}, \forall e \in E(H) : e_k \subsetneq e_j\} \cup \{e_j \in$

$E(H) : e_j \cap e_k = \emptyset\}$. The multilinear set of H_1 is equivalent to $Q = \{(a_1, \ldots, a_{|e_k|}, b) \in \{0,1\}^{|e_k|+1} \mid b = a_1 \cdots a_{|e_k|}\}$, which can be easily convexified by $|e_k|+1$ linear inequalities and $|e_k|$ nonnegativity constraints using the standard linearization (Rikun, 1997). The multilinear set of H_2 can be obtained from the multilinear set of H by substituting $\prod_{i \in e_k} x_i$ with y_j in all expressions, removing $\{x_i\}_{i \in e_k}$, and setting y_j to be a binary variable. Observe that, in this case, we did not need to introduce additional z variables. Therefore, this decomposition yields a set that can be easily convexified and another that corresponds to a strictly smaller hypergraph with $n - |e_k| + 1$ nodes. Moreover, H_2 is laminar because we removed e_k from $E(H)$ and replaced the nodes $v, \forall v \in e_k$ with e_k. Therefore, applying Theorem 6 on H_2 results in a decomposition of S into $\{S_t\}_{t=1}^{m+n}$, where the arborescence of S_t consists of a root e_t with children e_j for $j \in J_t$ and x_i for $i \in I_t$. Then, $S_t = \{(\boldsymbol{x_{I_t}}, \boldsymbol{y_{J_t}}, y_t) \in \{0,1\}^{|I_t|+|J_t|+1} \mid y_t = \prod_{i \in I_t} x_i \prod_{j \in J_t} y_j\}$. By convexifying each S_t using standard linearization techniques, we obtain the desired formulation for $\text{conv}(S)$. $\square$

Decomposability of a multilinear set.

Theorem 1 in Del Pia and Khajavirad (2018a) gives the following sufficient conditions for decomposability of the multilinear set S associated with a hypergraph H: if there exist $V_1, V_2 \subseteq V(H)$ such that $V_1 \cup V_2 = V(H)$, $H[V_1 \cap V_2]$ is complete, and either $e \subseteq V_1$ or $e \subseteq V_2$ holds for every $e \in E(H)$, then S is decomposable into the multilinear sets associated with $H[V_1]$ and $H[V_2]$. The conditions are equivalent to the special case of Theorem 6 with $I = V_1$ or V_2, and $\mathcal{F}'_I = \text{vert}(P_I)$. This is because the complete refinement of all multilinear terms in $V_1 \cap V_2$ is polynomial in the number of multilinear terms relating variables in $V_1 \cap V_2$.

We restate Theorem 1 in Del Pia and Khajavirad (2018a) in Corollary 2. We then provide a proof that uses Theorem 6 to show that S corresponds to a multilinear set over the Cartesian product of simplices that is decomposable.

Corollary 2 (Theorem 1 in Del Pia and Khajavirad (2018a))**.** *Let S be the multilinear set of a hypergraph H. Assume $V_1, V_2 \subsetneq V(H)$ are such that $V(H) = V_1 \cup V_2$, $E(H) = E(H[V_1]) \cup E(H[V_2])$, and $H[V_1 \cap V_2]$ is a complete hypergraph. Then, the convex hull of S is obtained by convexifying S_1 and S_2 separately where S_1 (resp. S_2) is the multilinear set of $H[V_1]$ (resp. $H[V_2]$).*

Proof. Let $E_1 = E(H[V_1])$ and $E_2 = E(H[V_2])$. We define the multilinear set over the Cartesian product of simplices corresponding to S. Let $P_v = [0,1]$, $\forall v \in V(H)$, $P = \prod_{v \in V(H)} P_v$, and $D = \mathrm{vert}(P)$. We denote by $F(e)$ the face corresponding to hyperedge $e \in E(H)$ (we convert x_v to $x_{v,1}$) and denote by $\mathcal{F}(E)$ the set of faces corresponding to hyperedges in E, *i.e.*, $\mathcal{F}(E) = \bigcup_{e \in E}\{F(e)\}$. Let $\mathcal{F} = \mathcal{F}(E(H)) \cup \mathcal{F}'$ where $\mathcal{F}' = \bigcup_{F \in \mathrm{vert}(P_{V_1 \cap V_2})} F \times P_{V(H)\setminus(V_1 \cap V_2)}$. The number of faces in $\mathcal{F}$ is a polynomial in $|V(H)|$ and $|E(H)|$ because $|\mathcal{F}| \leq |E(H)| + 2^{|V_1 \cap V_2|} \leq 2|E(H)| + |V(H)| + 1$ where the inequality holds because $|E(H)| \geq 2^{|V_1 \cap V_2|} - |V(H)| - 1$ as $H[V_1 \cap V_2]$ is complete. Now, we apply Theorem 6 to $\mathrm{ML}(D, \mathcal{F})$. Let $I = V_1$ and let $\bar{I} = V(H) \setminus V_1$. Let $\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2 = \texttt{partition}(\mathcal{F}, I)$. Then, $\mathcal{F}^0 = \mathcal{F}(E_1) \cup \mathcal{F}'$ and $\mathcal{F}^1 \cup \mathcal{F}^2 = \mathcal{F}(E_2 \setminus E_1)$. Let $\mathcal{F}^1_I = \bigcup_{F \in \mathcal{F}^1}\{F_I\}$ and let $\mathcal{F}'_I = \bigcup_{F \in \mathcal{F}'}\{F_I\}$. Set $\mathcal{F}'_I$ is a refinement of $\mathcal{F}^1_I$ because every face in $\mathcal{F}^1_I$ is in the form of $P_{V_1 \setminus V_2} \times F$ for some face F in $P_{V_1 \cap V_2}$ and $\mathcal{F}'_I = \bigcup_{F \in \mathrm{vert}(P_{V_1 \cap V_2})} P_{V_1 \setminus V_2} \times F$. For each $F \in \mathcal{F}^1_I$, let $C(F)$ be such that $C(F) \subseteq \mathcal{F}'_I$ and $\mathrm{conv}\bigl(\bigcup_{\bar{F} \in C(F)} \bar{F}\bigr) = F$, *i.e.*, $C(F)$ is the set of faces in $\mathcal{F}'$ that refine F. Therefore, $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}))$ can be obtained by convexifying $\mathrm{ML}(D_I, \mathcal{F}^0)$ and $\mathrm{ML}(D_{\bar{I}} \times \Delta^{2^{|V_1 \cap V_2|}}, \hat{\mathcal{F}}^1 \cup \hat{\mathcal{F}}^2)$ where $\hat{\mathcal{F}}^2 = \bigcup_{F \in \mathcal{F}^2}\{F_{V_2 \setminus V_1} \times \Delta^{2^{|V_1 \cap V_2|}}\}$ and $\hat{\mathcal{F}}^1 = \bigcup_{F \in \mathcal{F}^1}\{F_{V_2 \setminus V_1} \times \mathrm{conv}(\bigcup_{\bar{F} \in C(F)} e_{\mathrm{idx}(\bar{F})})\}$ where $e_i \in \{0,1\}^{2^{|V_1 \cap V_2|}}$ is the i^{th} principal vector. This is an equivalent decomposition because $\mathrm{ML}(D_I, \mathcal{F}^0)$ is the multilinear set over the variables corresponding to V_1 and $\mathrm{ML}(D_{V_2 \setminus V_1} \times \Delta^{2^{|V_1 \cap V_2|}}, \hat{\mathcal{F}}^1 \cup \hat{\mathcal{F}}^2\}$ is the multilinear set over the variables corresponding to V_2 where the variables corresponding to $V_1 \cap V_2$ is lifted to a space of $2^{|I|}$ variables. $\square$

Connection to Bienstock and Munoz (2018).

Bienstock and Munoz (2018) study formulations for the convex hull of S where the structural sparsity of S is described by a tree-decomposition. They show their formulations to be polynomially-sized when the tree-width is constant. Theorem 6 allows us to also derive formulations that are polynomial under these assumptions. For our set, the notion of tree-decomposition becomes

Definition 8. *Consider a set of proper faces $\mathcal{F}$ of $P := \prod_{v \in [n]} P_v$ where $P_v := \Delta^{K_v}$ for all $v \in [n]$. Let $D = \mathrm{vert}(P)$. A* tree-decomposition *of $\mathrm{ML}(D, \mathcal{F})$ is a pair (T, Q), where T is a tree and $Q = \{Q_t\}_{t \in V(T)}$ is a collection of subsets of $[n]$ (the indices of simplices) such that* *(i) For all $v \in [n]$, the set $\{t \in V(T) : v \in Q_t\}$ forms a subtree T_v of T,*

(ii) For each $F \in \mathcal{F}$, there is a $t \in V(T)$ such that $e(F) \subseteq Q_t$ where $e(F) := \{v \in [n] : J_F^v \subsetneq [K_v]\}\}$,

(iii) $\bigcup_{t \in V(T)} Q_t = [n]$.

The width *of a tree-decomposition is defined as $\max_{t\in V(T)} |Q_t| - 1$. The* treewidth *of $ML(D, \mathcal{F})$ is the minimum width over all tree-decompositions of $ML(D, \mathcal{F})$.*

It is known that, if the treewidth of $\mathrm{ML}(D, \mathcal{F})$ is a constant, then a tree-decomposition (T, Q) can be found in time linear in n and $|\mathcal{F}|$; see Bodlaender (1996).

Consider a set $\mathrm{ML}(D, \mathcal{F})$ for which we have obtained a tree-decomposition (T, Q). We can decompose $\mathrm{ML}(D, \mathcal{F})$ using Theorem 6 into $|V(T)|$ multilinear sets such that the convex hull of each set is described by a constant number of variables and constraints. This construction leads to

Corollary 3. *Consider a set of proper faces $\mathcal{F}$ of $P := \prod_{v\in[n]} P_v$ where $P_v := \Delta^{K_v}$ for all $v \in [n]$. Let $D = \mathrm{vert}(P)$. Assume that the treewidth of $ML(D, \mathcal{F})$ and that K_v for all $v \in [n]$ are all constants. Then, there exists an extended formulation for $\mathrm{conv}(ML(D, \mathcal{F}))$ whose size is polynomial in n and $|\mathcal{F}|$.*

Proof. Let r be the treewidth of $\mathrm{ML}(D, \mathcal{F})$. Let (T, Q) be a tree-decomposition of $\mathrm{ML}(D, \mathcal{F})$ obtained in linear time using the algorithm in (Bodlaender, 1996). It follows that the number of vertices in T is bounded above by a linear function in n and $|\mathcal{F}|$. Consider any edge $\{u, w\} \in E(T)$ such that $Q_u \cap Q_w = \emptyset$. Let $I_u = \bigcup_{t\in V_u} Q_t$ (resp. $I_w = \bigcup_{t\in V_w} Q_t$) where V_u (resp. V_w) is the set of vertices in $V(T)$ such that every vertex in V_u (resp. V_w) is reachable from u (resp. w) without using $\{u, w\}$. Sets I_u and I_w form a partition of $[n]$ by the definition of a tree-decomposition. Moreover, every face F in $\mathcal{F}$ can be expressed as either $F_{I_u} \times P_{I_w}$ or $P_{I_u} \times F_{I_w}$. Therefore, we can convexify $\mathrm{ML}(D, \mathcal{F})$ by convexifying $\mathrm{ML}(D_{I_u}, \mathcal{F}_{I_u})$ and $\mathrm{ML}(D_{I_w}, \mathcal{F}_{I_w})$ separately where $\mathcal{F}_{I_u} = \{F_{I_u} \mid F \in \mathcal{F}\}$ and $\mathcal{F}_{I_w} = \{F_{I_w} \mid F \in \mathcal{F}\}$. We may therefore assume that for every edge $\{u, w\} \in E(T)$, $Q_u \cap Q_w \neq \emptyset$. We next argue that we may assume that for every edge $\{u, w\} \in E(T)$, both $Q_u \subsetneq Q_w$ and $Q_w \subsetneq Q_u$ hold. Otherwise, we could shrink $\{u, w\}$ (*i.e.*, remove edge $\{u, w\}$, combine vertices u and w into one vertex v, and define $Q_v := Q_u \cup Q_w$) so as to obtain another tree-decomposition of $\mathrm{ML}(D, \mathcal{F})$ with fewer vertices. Recursively shrinking such edges would then lead to a tree-decomposition that satisfies the assumption.

Using Proposition 3, we show that there is a polynomially-sized formulation for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}))$ by constructing a refinement $\mathcal{F}'$ of $\mathcal{F}$ where $|\mathcal{F}'|$ is a linear function in n and $|\mathcal{F}|$. Define $\mathcal{F}' := \bigcup_{\{t\} \in V(T)} \mathcal{F}_t$ where $\mathcal{F}_t := \{F \times P_{[n]\setminus Q_t}\}_{F \in \mathrm{vert}(P_{Q_t})}$ for all $t \in V(T)$. We first argue that $\mathcal{F}'$ is a refinement of $\mathcal{F}$. For every $F \in \mathcal{F}$, there is $t \in V(T)$ such that $F = F_{Q_t} \times P_{[n]\setminus Q_t}$ and F_{Q_t} is a face of P_{Q_t} by (ii) in the definition of tree-decomposition. Since for every vertex v of P_{Q_t}, $v \times P_{[n]\setminus Q_t}$ is in $\mathcal{F}'$, it follows that $\mathcal{F}'$ refines F, therefore, $\mathcal{F}'$ refines $\mathcal{F}$. We next argue that $|\mathcal{F}'|$ is bounded above by a linear function of n and $|\mathcal{F}|$. By the definition of $\mathcal{F}'$, we have $|\mathcal{F}'| \leq \sum_{t \in V(T)} \prod_{v \in Q_t} K_v$. The right-hand-side of this expression is a linear function of n and $|\mathcal{F}|$ because (i) $\{K_v\}_{v \in [n]}$ are constants by assumption, (ii) the maximum cardinality of Q_t for $t \in V(T)$ is bounded above by the treewidth of $\mathrm{ML}(D, \mathcal{F})$ plus one, which is constant, and (iii) $|V(T)|$ is a linear function of n and $|\mathcal{F}|$. Therefore, $\mathcal{F}'$ is a refinement of $\mathcal{F}$ where $|\mathcal{F}'|$ is a linear function of n and $|\mathcal{F}|$.

Next, we obtain a formulation for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}'))$. We apply Theorem 6 to decompose $\mathrm{ML}(D, \mathcal{F}')$. Pick an edge $\{u, w\}$ in $E(T)$ where w is a leaf. Let $I = \bigcup_{t \in V(T): t \neq w} Q_t$. It follows from the assumption that $Q_w \subsetneq Q_u$ and from (i) in the definition of a tree-decomposition that $I \subsetneq [n]$. Let $\mathcal{F}^0, \mathcal{F}^1, \mathcal{F}^2 = \texttt{partition}(\mathcal{F}', I)$. We argue that $\mathcal{F}^0 = \mathcal{F}' \setminus \mathcal{F}_w$, $\mathcal{F}^1 = \mathcal{F}_w$, and $\mathcal{F}^2 = \emptyset$. Recall that $\mathcal{F}' = \bigcup_{t \in V(T)} \mathcal{F}_t$. For $t \in V(T)$ that is distinct from w, $\mathcal{F}_t \subseteq \mathcal{F}^0$ since $Q_t \subseteq I$. For $F \in \mathcal{F}_w$, $F \in \mathcal{F}^1$ since F is a vertex of P_{Q_w}, $Q_w \cap I \neq \emptyset$, and $Q_w \cap \bar{I} \neq \emptyset$. Therefore, $\mathcal{F}^0 = \mathcal{F}' \setminus \mathcal{F}_w$, $\mathcal{F}^1 = \mathcal{F}_w$, and $\mathcal{F}^2 = \emptyset$. Let $\mathcal{F}^i_I = \{F_I \mid F \in \mathcal{F}^i\}$ for $i \in \{0, 1\}$. We argue that $\mathcal{F}^0_I$ is a refinement of $\mathcal{F}^1_I$. Since $\mathcal{F}^1 = \mathcal{F}_w$, then $\mathcal{F}^1_I = \{F_{I \cap Q_w} \times P_{I \setminus Q_w}\}_{F \in \mathrm{vert}(P_{Q_w})} = \{F \times P_{I \setminus Q_w}\}_{F \in \mathrm{vert}(P_{Q_w \cap I})}$. Also, since $\mathcal{F}_u \subseteq \mathcal{F}^0$, then $\mathcal{F}^0_I \supseteq \{F \times P_{I \setminus Q_u}\}_{F \in \mathrm{vert}(P_{Q_u})}$. Since $Q_w \cap I = Q_w \cap Q_u \subseteq Q_u$, then $\mathcal{F}^0_I$ is a refinement of $\mathcal{F}^1_I$. Therefore, we can apply Theorem 6 to decompose $\mathrm{ML}(D, \mathcal{F}')$ without adding any additional faces. Let $\mathrm{idx}(F)$ be the index of F in $\mathcal{F}_u$ for every $F \in \mathcal{F}_u$. Let $e_i \in \{0, 1\}^{|\mathcal{F}_u|}$ be the i^{th} principal vector. Let $C(F)$ be such that $C(F) \subseteq \mathcal{F}_u$ and $\mathrm{conv}\left(\bigcup_{\bar{F} \in C(F)} \bar{F}_I\right) = F$ for $F \in \mathcal{F}^1_I$. By Theorem 6, we can obtain a formulation for $\mathrm{conv}(\mathrm{ML}(D, \mathcal{F}'))$ by separately convexifying $\mathrm{ML}(D_I, \mathcal{F}^0_I)$ and $\mathrm{ML}(D_{\bar{I}} \times \Delta^{|\mathcal{F}_u|}_{0,1}, \hat{\mathcal{F}}^1)$ where $\hat{\mathcal{F}}^1 = \left\{F_{\bar{I}} \times \mathrm{conv}\left(\bigcup_{\bar{F} \in C(F)} \{e_{\mathrm{idx}(\bar{F})}\}\right)\right\}_{F \in \mathcal{F}^1}$. The latter set is a multilinear set over the Cartesian product of a constant number of simplices with a constant number of faces. The former set either is of the same form as the latter (when it corresponds to a tree with a single node) or is a multilinear set represented by a constant-treewidth tree-decomposition with $V(T)-1$ vertices, *i.e.*, the tree-decomposition obtained by removing edge $\{u,w\}$ from T. Therefore, by recursively applying Theorem 6, we can convexify $\mathrm{ML}(D,\mathcal{F}')$ by separately convexifying $|V(T)|$ multilinear sets, for which constant-size convex hulls descriptions can be explicitly constructed since every multilinear set is associated with a constant number of simplices and a constant number of faces. This construction yields a polynomially-sized formulation for $\mathrm{ML}(D,\mathcal{F}')$. In turn, this formulation can be used to obtain a polynomially-sized formulation for $\mathrm{ML}(D,\mathcal{F})$. □

The proof of Corollary 3 recursively applies the idea that for each leaf-node, if we track all possible multilinear terms that relate to variables in the neighboring node, then the convex hull can be developed independent of the rest of the tree. This decomposition follows from Theorem 6 since the convex hull of all multilinear terms for a Cartesian product of simplices is a simplex. It follows that the main burden in the explicit construction of this convex hull resides in the determination of the tree-decomposition, which can be found in time linear in n and $|\mathcal{F}|$.

Theorem 3.5 in Bienstock and Munoz (2018) can then be viewed as a special case of Corollary 3.

An Example Showing Strict Generalization.

Example 6 shows that the framework for convexification presented in this chapter is in fact strictly more general than the results of the literature reviewed above. It does so by describing a set whose convex hull cannot be directly obtained using these results but can be derived from Theorem 6, together with other theorems in this chapter.

Example 6. *Consider*

$$S=\left\{(\boldsymbol{x},\boldsymbol{y})\in\{0,1\}^{2n}\times\{0,1\}^{3n}\;\middle|\;\begin{array}{ll} y_j=(1-x_j)\prod_{i=1}^{j-1}x_i, & \forall j\in[n]\\ y_{n+j}=y_jx_{n+j}, & \forall j\in[n]\\ y_{2n+j}=y_jx_{n+j}x_{n+j+1}, & \forall j\in[n-1]\\ y_{3n}=y_nx_{2n}x_{n+1} & \end{array}\right\}.$$

Theorem 6 can be used to decompose S into S_1 and S_2 where S_1 is the multilinear *set over $x_1, \ldots, x_n$ with multilinear terms $y_1, \ldots, y_n$ and S_2 is the multilinear set over $x_{n+1}, \ldots, x_{2n}, y_1, \ldots, y_n$ with multilinear terms $y_{n+1}, \ldots, y_{3n}$ using $I = [n]$. Specifically,*

$$S_1 = \left\{ (\boldsymbol{x_I}, \boldsymbol{y_I}) \in \{0,1\}^n \times \{0,1\}^n \;\middle|\; y_j = (1 - x_j) \prod_{i=1}^{j-1} x_i, \;\forall j \in [n] \right\}$$

$$S_2 = \left\{ (\boldsymbol{x_{\bar{I}}}, \boldsymbol{y_I}, \boldsymbol{y_{\bar{I}}}) \in \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^{2n} \;\middle|\; \begin{array}{ll} y_{n+j} = y_j x_{n+j}, & \forall j \in [n] \\ y_{2n+j} = y_j x_{n+j} x_{n+j+1}, & \forall j \in [n-1] \\ y_{3n} = y_n x_{2n} x_{n+1} & \\ \mathbf{1}^\intercal \boldsymbol{y_I} \le 1 & \end{array} \right\}.$$

We remark that if we replace y_j for $j \in [n]$ with $\bar{y}_j = \prod_{i=1}^{j} x_i$, the resulting set can still be convexified using the techniques in this chapter. This is because, as mentioned before, S_1 is an affine transform of the set obtained after replacing y_j. We can obtain conv(S) *by convexifying S_1 and S_2 separately. A polynomially-sized formulation for* conv(S_1) *can be obtained using Theorem 3 because $\boldsymbol{y_I}$ corresponds to a facial decomposition of the Cartesian product of simplices corresponding to $\boldsymbol{x_I}$ and because it has the adjacency property. We can obtain a polynomially-sized formulation for* conv(S_2) *because the set of faces corresponding to $\{y_j\}_{j=n+1}^{3n}$ is poly-completable according to Theorem 5.*

Existing results in the literature do not apply to S. The treewidth of S is at least $n+2$ because the degree of y_{3n} in terms of $\boldsymbol{x}$ variables is $n+2$. Hence, Theorem 3.5 in Bienstock and Munoz (2018) does not yield a polynomially-sized formulation. Consider next the hypergraph H associated with S obtained by creating hyperedges for all monomials that appear in the expression: $V(H) = [2n]$ and $E(H) = E_1 \cup E_2 \cup E_3$ where

$$\begin{aligned} E_1 =& \{\{1,2\}, \{1,2,3\}, \ldots, \{1,2,\ldots,n\}\}, \\ E_2 =& \{\{1,n+1\}, \{\{1,2,n+2\}, \ldots, \{1,2,\ldots,n,2n\}\}, \\ E_3 =& \{\{1,n+1,n+2\}, \{1,2,n+2,n+3\}, \ldots, \\ & \{1,\ldots,n-1,2n-1,2n\}, \{1,\ldots,n+1,2n\}\}. \end{aligned}$$

Sets E_1, E_2, and E_3 are associated with multilinear terms $\{y_j\}_{j\in[n]}$, $\{y_{n+j}\}_{j\in[n]}$, and $\{y_{2n+j}\}_{j\in[n]}$, respectively. The hypergraph H is neither laminar nor γ-acyclic. In fact, H contains many γ-cycles in H, for example, $v_1 - e_1 - v_2 - e_2 - v_3 - e_3 - v_1$ where $v_1 = 1$,

$v_2 = k$, $v_3 = n+k$, $e_1 = \{1,\dots,k\}$, $e_2 = \{1,\dots,k,n+k\}$, $e_3 = \{1,\dots,k-1,n+k-1,n+k\}$ for all $k = 2,\dots,n$ and $v_1 - e_1 - v_2 - e_n - \dots - v_n - e_n - v_1$ where $v_j = n+j$ for all $j \in [n]$, $e_j = \{1,\dots,j,n+j,n+j+1\}$ for all $j \in [n-1]$, and $e_n = \{1,\dots,n,n+1,2n\}$. Therefore, we cannot apply either Theorem 10 or Corollary 18 from Del Pia and Khajavirad (2018b) to convexify S. Further, decomposition Theorem 1 in Del Pia and Khajavirad (2018a) cannot be used to convexify S. In fact, $H[V']$ is complete for $V' \subseteq V(H)$ only if $V' = \{1,2\}$ or $|V'| = 1$. However, there is no $V_1, V_2 \subseteq V(H)$ such that $V_1 \cap V_2 = \{1,2\}$ or $|V_1 \cap V_2| = 1$, and either $e \subseteq V_1$ or $e \subseteq V_2$ holds for every $e \in E(H)$.

2.6 Applications

In this section, we apply our results to build improved formulations for specific optimization problems. In Section 2.6.1, we build an improved model for a standard decision tree using the results in Section 2.5. In Section 2.6.2, we perform computational experiments that establish the benefits of our new formulations. In Section 2.6.3, we propose a new predictive model, *multilinear decision trees*, which outperforms classical models in approximating various nonlinear functions.

2.6.1 Improved Formulations for TEO

In this section, we provide mixed-integer linear programming (MILP) formulations for TEO that are provably tighter than those in the literature. By Remark 1, the leaves in a decision tree correspond to a facial decomposition. We can therefore rewrite TEO as

$$\max\left\{\sum_{t\in[T]}\sum_{\ell\in\textbf{leaves}(t)} p_{t,\ell}y_{t,\ell}\;\middle|\;(\boldsymbol{x},\boldsymbol{y_t})\in \mathrm{ML}(D,\mathcal{F}_t), \forall t\in[T]\right\},$$

where $D = \prod_{v\in[n]} \Delta_{0,1}^{K_v}$ and for all $t \in [T]$, $\mathcal{F}_t$ is the collection of faces corresponding to $y_{t,\ell}$ for all $\ell \in \textbf{leaves}(t)$. The MILP formulation we propose uses the constraints of the convex hull of $\mathrm{ML}(D,\mathcal{F}_t)$, known because of Proposition 1, for each $t \in [T]$ to obtain

$$\max \sum_{t\in[T]}\sum_{\ell\in\textbf{leaves}(t)} p_{t,\ell}y_{t,\ell} \tag{2.25a}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathbf{leaves}(t): J^v_{t,\ell} \subseteq J} y_{t,\ell} \leq \sum_{j \in J} x_{v,j}, \qquad \forall t \in [T],\ \forall v \in [n],\ \forall \emptyset \neq J \subseteq [K_v], \tag{2.25b}$$

$$\boldsymbol{x_v} \in \Delta^{K_v}_{0,1}, \qquad \forall v \in [n], \tag{2.25c}$$

$$\boldsymbol{y_t} \in \Delta^{|\mathbf{leaves}(t)|}, \qquad \forall t \in [T]. \tag{2.25d}$$

The size of this formulation is exponential in $\{K_v\}_{\forall v \in [n]}$. However, as we discussed in Section 2.5.1, there is a polynomial-time separation algorithm for (2.25b); see Proposition 6. Alternatively, an extended polynomially-sized formulation follows directly from Theorem 2.

Corollary 4. *Formulation (2.25) is integral and defines the convex hull of the problem when $T = 1$.*

It follows from Corollary 4 that formulation (2.25) is tightest possible if we focus on each tree separately. Thus, the LP relaxation of (2.25) is tighter than that of Mišić (2020), which models each $\text{ML}(D, \mathcal{F}_t)$ with a system that does not describe its convex hull. We generate TEOs using various combination of real data sets, ensemble models, and number of trees and compare the LP relaxation objective values of two formulations. To compare the tightness of these formulations, we use *relative LP gap*. Given an optimization problem instance and a formulation f, we denote by δ_f the absolute deviation between the optimal objective value and the LP relaxation objective value. The relative LP gap of f with base formulation f_b is computed as $\frac{\delta_f}{\delta_{f_b}}$. Table 2.1 displays relative LP gaps of formulation (2) in Mišić (2020) where formulation (2.25) is used to as the basis of comparison. It shows that the relative LP gaps of formulation (2) in Mišić (2020) are between 1.20 and 4.65. Our formulation also has the advantage of only requiring a collection of faces, and not the decision tree associated with the facial set.

When every input variable of the tree ensemble is numerical, the faces corresponding to the leaves satisfy the adjacency property. Therefore, by Theorem 3, we can obtain a formulation that is polynomially-sized in the input parameters (T, n, and $K_v, \forall v \in [n]$) by replacing (2.25b) with

$$\sum_{\ell \in \mathbf{leaves}(t): J^v_{t,\ell} \subseteq [a..b]} y_{t,\ell} \leq \sum_{j=a}^{b} x_{v,j}, \qquad \forall t \in [T],\ \forall v \in [n],\ \forall a, b \subseteq [K_v] : a \leq b. \tag{2.26}$$

Table 2.1: Relative LP gap of formulation (b) which is (2) in Mišić (2020) with our formulation (2.25) denoted as (a).

Dataset	Ensemble model	T	Relative LP gap	
			(a)	(b)
Concrete strength (Yeh, 1998)	Random forest	100	1.00	3.95
		300	1.00	4.65
		500	1.00	4.44
	Boosted trees	100	1.00	4.57
		300	1.00	1.79
		500	1.00	1.76
Wine quality (Cortez et al., 2009)	Random forest	100	1.00	1.20
		200	1.00	1.20
		300	1.00	1.20
	Boosted trees	100	1.00	1.76
		200	1.00	1.71
		300	1.00	1.64

This special case has been studied. In Kim et al. (2019), we described an equivalent formulation when all the input variables are numerical and showed its tightness. Chen and Mišić (2021) considered a special case where all input variables are binary and show the tightness of this formulation. Formulation (2.25) and Corollary 4 generalize these results.

2.6.2 Constrained TEO

In this section, we consider TEO with additional constraints and compare the computational performance of the formulations we propose with existing formulations for piecewise-linear functions from the literature. To do so, we consider instances of multi-commodity transportation problems generated in a manner similar to Vielma et al. (2010) except that the cost function is described as a tree ensemble. The transportation problem is defined over the complete bipartite graph $G = (U, V, E)$, where U (resp. V) refers to the set of the supply (resp. demand) nodes and E refers to the set of arcs. For each commodity $i \in [C]$ where C is the number of commodities, a supply (resp. demand) amount $s_{u,i}$ (resp. $d_{v,i}$) is assigned to each supply (resp. demand) node $u \in U$ (resp. $v \in V$). An individual capacity denoted by $c_{e,i}$ is generated for each arc $e \in E$ and for each commodity $i \in [C]$. We introduce variable $z_{e,i} \in \mathbb{R}$ to represent the flow of commodity

Table 2.2: Size and tightness of formulations for a single decision tree with n input variables and m leaves. Columns `nBinVar`, `nConVar`, and `nConstr` display the numbers of binary variables, continuous variables, and constraints, respectively. Column `Integrality` indicates whether every extreme point of the LP relaxation is integral or not.

Formulations	`nBinVar`	`nConVar`	`nConstr`	`Integrality`
(TEO)	$m-1$	$\mathcal{O}(nm)$	$\mathcal{O}(nm^2)$	yes
(TEOM)	$m-1$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	no
(MC)	$m-1$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	yes
(DCC)	$m-1$	$\mathcal{O}(m2^n)$	$\mathcal{O}(m)$	yes
(DLog)	$\lceil \log_2 m \rceil$	$\mathcal{O}(m2^n)$	$\mathcal{O}(m)$	yes

$i \in [C]$ on arc $e \in E$. The transportation cost incurred on arc $e \in E$ is given as the sum of the values of pre-trained decision trees defined over common independent variables, $\boldsymbol{z_e} = (z_{e,1}, \ldots, z_{e,C})$, and denoted as $f_e(\boldsymbol{z_e}) = \sum_{t \in [T_e]} f_{e,t}(\boldsymbol{z_e})$ where T_e is the number of trees describing the cost function on arc e, and $f_{e,t}(\boldsymbol{z_e})$ is the t^{th} decision tree function for arc $e \in E$. The objective function is the sum of independent arc costs, $\sum_{e \in E} f_e(\boldsymbol{z_e})$. This problem is a constrained TEO where we minimize the sum of values of pre-trained decision trees under balance and capacity constraints on the input variables $\boldsymbol{z}$.

We compare our formulation (TEO) with four other formulations – (TEOM), (MC), (DCC), and (DLog) – from the literature. Clearly, f_e is modeled as a sum of piecewise constant functions, one for each tree t in the ensemble that models f_e. Such a problem can be formulated using models for piecewise-constant functions. However, our formulations will directly use the tree ensemble representation of f_e. The explicit formulations are given in 2.6.5. Each formulation models f_e independently of $f_{e'}$ for distinct arcs $e, e' \in E$. From now, we fix e. (TEO), the formulation we propose, is obtained by formulating $(\boldsymbol{x_e}, \boldsymbol{y_{e,t}}) \in \mathrm{ML}(D_e, \mathcal{F}_{e,t})$ for $t \in [T_e]$, where D_e and $\{\mathcal{F}_{e,t}\}_{t \in [T_e]}$ are defined from the pre-trained tree ensemble model for f_e. Variable $\boldsymbol{y_{e,t}}$ indicates whether the corresponding leaf is active for tree $t \in [T_e]$. Variable $\boldsymbol{x_e}$ indicates the hyper-rectangle that contains $\boldsymbol{z_e}$ and is contained in the region corresponding to each active leaf. In (TEO), for each $(\boldsymbol{x_e}, \boldsymbol{y_{e,t}}) \in \mathrm{ML}(D_e, \mathcal{F}_{e,t})$, where $t \in [T]$, we include the constraints in Theorem 3 describing $\mathrm{conv}(\mathrm{ML}(D_e, \mathcal{F}_{e,t}))$ except redundant ones identified in Proposition 2. (TEOM) is a formulation for TEO problems, which extends formulation (2) for a tree ensemble proposed in Mišić (2020) in a manner similar to Mistry et al. (2021), by

Table 2.3: Average relative LP gaps for N multi-commodity transportation problem instances with (TEO) as the basis for comparison. The smallest average value(s) are in bold for each row.

C	D	T	N	(TEO)	(TEOM)	(MC)	(DCC)	(DLog)
2	4	1	10	**1.00**	7.47	**1.00**	**1.00**	**1.00**
2	4	3	10	**1.00**	8.28	5.18	5.18	5.18
2	4	5	10	**1.00**	9.86	8.19	8.19	8.19
2	4	7	10	**1.00**	11.99	11.51	11.51	11.51
2	4	9	10	**1.00**	13.42	13.85	13.85	13.85
2	5	1	10	**1.00**	11.67	**1.00**	**1.00**	**1.00**
2	5	3	10	**1.00**	18.07	5.95	5.95	5.95
2	5	5	10	**1.00**	17.23	7.70	7.70	7.70
2	5	7	10	**1.00**	15.47	8.13	8.13	8.13
2	5	9	10	**1.00**	16.81	9.53	9.53	9.53
3	4	1	10	**1.00**	10.38	**1.00**	**1.00**	**1.00**
3	4	3	10	**1.00**	7.78	1.62	1.62	1.62
3	4	5	10	**1.00**	8.40	1.98	1.98	1.98
3	4	7	10	**1.00**	8.52	2.18	2.18	2.18
3	4	9	10	**1.00**	9.30	2.46	2.46	2.46
3	5	1	10	**1.00**	22.97	**1.00**	**1.00**	**1.00**
3	5	3	10	**1.00**	22.51	2.57	2.57	2.57
3	5	5	10	**1.00**	30.66	4.08	4.08	4.08
3	5	7	10	**1.00**	26.64	4.05	4.05	4.05
3	5	9	10	**1.00**	27.03	4.33	4.33	4.33
Total average			200	**1.00**	15.22	4.87	4.87	4.87

connecting $\boldsymbol{z}_e$ variables to the variables in the tree ensemble model. The difference between (TEO) and (TEOM) is that (TEO) describes $\text{conv}(\text{ML}(D_e, \mathcal{F}_{e,t}))$ but (TEOM) does not. It follows that the LP relaxation of (TEO) is tighter than that of (TEOM). This is confirmed by our experiments with the formulations reported in Table 2.3 where we find that the relative performance of (TEOM) worsens as the depth of trees in the ensemble increases.

The next three formulations –(MC), (DCC), and (DLog)– are based on general modeling techniques for piecewise-linear functions. Let $m_{e,t}$ be the number of leaves in the t^{th} tree in the ensemble model corresponding to arc e. Both (MC) and (DCC) (see Vielma (2015) for details) include a disjunctive constraint $\boldsymbol{y} \in \Delta_{0,1}^{m_{e,t}}$, where y_j indicates whether point $(z_{e,1}, \ldots, z_{e,C})$ is contained in the hyper-rectangle corresponding to the

j^{th} leaf. They differ in the way they describe pieces: (MC) uses an $\mathcal{H}$-representation of the pieces whereas (DCC) uses their $\mathcal{V}$-representation. Formulation (DLog) (as in Ibaraki (1976); Vielma and Nemhauser (2011)) uses a $\mathcal{V}$-representation similar to DCC except that it only requires $\lceil \log_2 m_{e,t} \rceil$ binary variables to express the choice among the $m_{e,t}$ pieces.

Next we compare, in Table 2.2, the size and tightness of formulations for a constrained TEO with a single decision tree with n independent variables and m leaves. (TEO), (TEOM), and (MC) are polynomial-sized formulations, whereas the number of continuous variables in (DCC) and (DLog) is exponential in n. We remark that (TEO), (MC), (DCC), and (DLog) capture the convex hull of a single decision tree. For problems involving many trees, the bound from (TEO) is the tightest and we use it as the basis of comparison in Table 2.3.

We generate two- and three-commodity transportation problem instances, *i.e.*, $C \in \{2, 3\}$ on a complete bipartite graph with five supply nodes and two demand nodes. We construct a collection of instances by varying the tree ensemble training parameters T and D, where T is the number of trees used to describe the cost on an arc $e \in E$ and D is the maximum depth of those trees. It follows that each instance corresponds to a constrained TEO with $10T$ trees. Given (C, D, T), we construct an instance as follows. First, we randomly generate supply, demand, and capacities. Second, for each arc $e \in E$, we define a nonlinear hidden cost function g_e, generate sample data points describing g_e, and train a random forest model using those data points. Details of the procedure used to generate the instances are provided in 2.6.4. For each (C, T, D), we randomly construct 10 instances. All five formulations are tested on 200 instances with $C \in \{2, 3\}$, $D \in \{4, 5\}$, and $T \in \{1, 3, 5, 7, 9\}$. Since (TEO) and (TEOM) outperform the remaining formulations, we generate 20 hard instances with $(C, D, T) \in \{(2, 6, 20), (3, 6, 20)\}$ and solve them using these formulations.

We perform our computational experiments on a computer running Linux Mint 19.3 with Intel i7-6700K CPU cores running at 4.00GHz and 48GB of memory. The code is written in Julia v1.6.3 with Gurobi v9.0.3 (Gurobi Optimization, LLC, 2021) as an MIP solver and JuMP package v0.21.10 (Dunning et al., 2017). We set a time limit of one hour for the 200 smaller instances and three hours for the remaining instances. Table 2.3 compares the tightness of formulations. When $T = 1$, the LP relaxation values

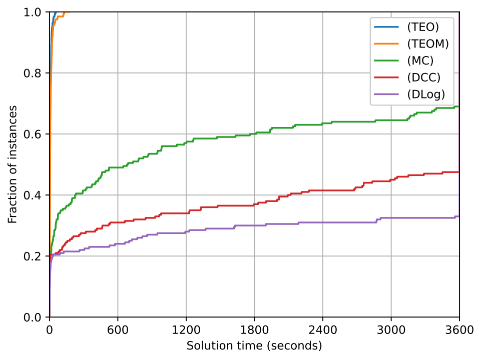
Table 2.4: Averages solution times for N multi-commodity transportation problem instances. The number of instances that reach one hour time limit is given as superscript. The smallest average value(s) are in bold for each row.

C	D	T	N	(TEO)	(TEOM)	(MC)	(DCC)	(DLog)
2	4	1	10	0.7	**0.7**	**0.7**	0.8	1.2
2	4	3	10	2.5	**2.4**	46.1	429.5	1880.8^{3}
2	4	5	10	**3.3**	3.5	608.4	3218.8^{7}	3600.0^{10}
2	4	7	10	**4.5**	4.8	2959.2^{3}	3600.0^{10}	3600.0^{10}
2	4	9	10	**4.1**	5.3	3536.3^{8}	3600.0^{10}	3600.0^{10}
2	5	1	10	2.6	2.5	**2.2**	2.5	6.0
2	5	3	10	**4.6**	6.9	578.5	3098.2^{6}	3600.0^{10}
2	5	5	10	**10.0**	14.1	3555.9^{9}	3600.0^{10}	3600.0^{10}
2	5	7	10	**17.7**	34.7	3600.0^{10}	3600.0^{10}	3600.0^{10}
2	5	9	10	**31.8**	73.6	3600.0^{10}	3600.0^{10}	3600.0^{10}
3	4	1	10	**0.8**	1.0	1.1	1.5	2.3
3	4	3	10	**3.0**	3.6	24.0	146.1	432.0
3	4	5	10	**3.8**	5.8	145.5	1681.1^{1}	3140.1^{5}
3	4	7	10	**7.1**	7.3	657.4	2861.1^{3}	3600.0^{10}
3	4	9	10	9.0	**8.0**	1457.6	3600.0^{10}	3600.0^{10}
3	5	1	10	2.3	**2.0**	2.1	4.7	14.4
3	5	3	10	**6.0**	8.2	483.7	1662.8	3051.3^{7}
3	5	5	10	**10.5**	12.5	2446.2^{3}	3523.4^{9}	3600.0^{10}
3	5	7	10	**13.6**	19.3	3600.0^{10}	3600.0^{10}	3600.0^{10}
3	5	9	10	**13.4**	19.8	3600.0^{10}	3600.0^{10}	3600.0^{10}
Total average			200	**7.6**	11.8	1545.2	2271.5	2586.4

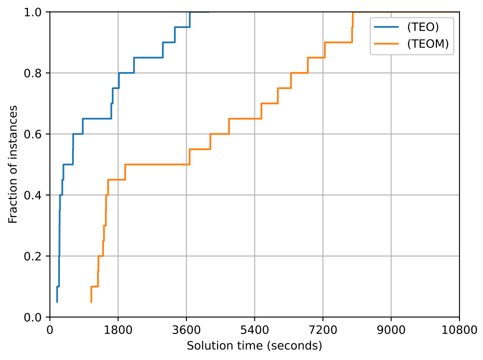
Table 2.5: Average relative LP gaps (with (TEO) as the basis for comparison) and solution times for N multi-commodity transportation problem instances. The number of instances that reach the three hours time limit is given as superscript. The smallest relative LP gap and average value are in bold for each row.

C	D	T	N	Relative LP gap		Solution time	
				(TEO)	(TEOM)	(TEO)	(TEOM)
2	6	20	10	**1.00**	25.03	**2293.8**	6538.5^{1}
3	6	20	10	**1.00**	68.59	**300.7**	1896.5

of (TEO), (MC), (DCC), and (DLog) are the same and tighter than (TEOM) because those formulations describe the convex hull for a single tree. However, when $T > 1$, the LP relaxation values of (TEO) are tighter than those of other formulations. This is



(a) 200 instances.



(b) 20 hard instances.

Figure 2.9: Performance profiles of solution times. The horizontal axis represents the solution time and the vertical axis represents the fraction of instances solved. Figure 2.9a displays the performance profile of 200 instances where $C \in \{2, 3\}$, $D \in \{4, 5\}$, and $T \in \{1, 3, 5, 7, 9\}$. Figure 2.9b displays the performance profile of 20 hard instances where $C \in \{2, 3\}$, $D = 6$, and $T = 20$.

because, in (TEO), the formulations for various trees that describe the cost on the same arc are connected by $\boldsymbol{x}$ variables, while these variables are not present in (MC), (DCC), and (DLog). The instances are classified using (C, D, T) and the average time taken by each formulation over 10 instances of each problem type is reported in Table 2.4 with selected records. Clearly, (TEO) and (TEOM) outperform the other three formulations when $T > 1$. Moreover, (TEO) performs better than (TEOM) as T increases. We perform additional experiments only with (TEO) and (TEOM) with larger maximum depth ($D = 6$) and number of trees ($T = 20$). The results are summarized in Table 2.5, which shows that (TEO) performs better than (TEOM) on hard instances. A graphical vizualization of the results is presented in Figure 2.9 using performance profiles (Dolan and Moré, 2002). In conclusion, we find that our new modeling framework outperforms existing formulations when modeling tree ensembles.

2.6.3 Generalized Decision Trees

Although TEO problems and multilinear optimization problems are deeply connected with one another, decision trees that predict a constant value at each node cannot model general multilinear functions. A possibly remedy is to perform regression with all interaction terms at each leaf node. However, this adds a significant number of parameters with the potential for over-fitting. In this section, we propose a generalization of decision trees where each tree can exactly represent a multilinear function, but the flexibility is limited by penalizing discontinuities at the boundaries. We show that this way of constructing decision trees provides a better fit for many nonlinear functions with/without random noise. In this way, we expand the expressiveness of decision trees without adversely affecting their prediction performance. With this modification, every multilinear problem can now be expressed as a generalized decision tree problem and vice-versa. We remark that the primary technique to solve TEO problems with these generalized decision trees would be to construct tight relaxations (Crama, 1993; Rikun, 1997; Sherali, 1997; Tawarmalani and Sahinidis, 2002; Luedtke et al., 2012; He and Tawarmalani, 2021). Hence, the framework we have developed would also be useful for constructing relaxations for such TEO problems.

Formally, a *generalized decision tree* (G-tree) is a piecewise-nonlinear function $f(X;T,\mathcal{G}) : \mathbb{R}^n \mapsto \mathbb{R}$, where T is a decision tree and $\mathcal{G} = \{g_\ell\}_{\ell \in \mathbf{leaves}(T)}$ is a collection of nonlinear functions defined over the hyper-rectangles corresponding to the leaves of T. Given a G-tree with T and $\mathcal{G}$, the evaluation of point $\bar{X} \in \mathbb{R}^n$ is computed as $g_{\ell(\bar{X})}(\bar{X})$ where $\ell(X)$ is the leaf $\ell \in \mathbf{leaves}(T)$ corresponding to X. A standard decision tree is a G-tree with a collection of constant functions $\mathcal{G}$. In this chapter, we consider *multilinear decision trees* (ML-trees), where $g_\ell(X)$ is multilinear in X for all $\ell \in \mathbf{leaves}(T)$.

We show that even a simple two-step algorithm for training an ML-tree generates a model that fits the data generated by sampling nonlinear functions (with error) better than classical decision tree models. Given data points $\{(\bar{\boldsymbol{x}}^{\boldsymbol{i}}, \bar{y}^i)\}_{i=1}^{n_{\text{data}}}$, we first train a decision tree T using any decision tree training algorithm such as CART and, second, we find the best fitting g_ℓ for all $\ell \in \mathbf{leaves}(T)$. We next expand on the second step. Let Q_ℓ be the hyper-rectangle that corresponds to leaf ℓ. To define $\mathcal{G}$ for an ML-tree, it is sufficient to specify the function value for every vertex of every hyper-rectangle Q_ℓ; see 2.6.6. Let $u_{\ell,k}$ be the function value of the k^{th} vertex of Q_ℓ for $\ell \in \mathbf{leaves}(T)$ and $k \in [2^n]$. Let $g(\boldsymbol{x}; Q, \boldsymbol{s})$ be the unique multilinear function of $\boldsymbol{x}$ that achieves the values $\boldsymbol{s} \in \mathbb{R}^{2^n}$ at the corner points of hyper-rectangle Q. Let $W = \bigcup_{\ell \in \mathbf{leaves}(T)} \text{vert}(Q_\ell) = \{t_1, \ldots, t_{|W|}\}$

and let $\text{idx}(\ell, k)$ be the index of the k^{th} vertex of Q_ℓ in W. Given nonnegative regularization weight parameters $\alpha, \beta \in \mathbb{R}_+$, the following optimization problem determines $\{u_{\ell,k}\}_{\ell \in \textbf{leaves}(t), k \in [2^n]}$:

$$\min_{\boldsymbol{u},\boldsymbol{\rho},\boldsymbol{z},\boldsymbol{w}} \quad \rho_{\text{trn}} + \alpha \cdot \rho_{\text{intra}} + \beta \cdot \rho_{\text{inter}} \tag{2.27a}$$

$$\text{s.t.} \quad \rho_{\text{trn}} \geq \frac{1}{n_{\text{data}}} \sum_{i=1}^{n_{\text{data}}} \left(\bar{y}^i - g(\bar{\boldsymbol{x}}^{\boldsymbol{i}}; Q_{\ell(\bar{\boldsymbol{x}}^i)}, \boldsymbol{u}_{\boldsymbol{\ell}(\bar{\boldsymbol{x}}^i)}) \right)^2, \tag{2.27b}$$

$$\rho_{\text{intra}} \geq \frac{1}{|\textbf{leaves}(T)| 2^n} \sum_{\ell \in \textbf{leaves}(T)} \sum_{k \in [2^n]} (u_{\ell,k} - z_\ell)^2, \tag{2.27c}$$

$$\rho_{\text{inter}} \geq \frac{1}{|\textbf{leaves}(T)| 2^n} \sum_{\ell \in \textbf{leaves}(T)} \sum_{k \in [2^n]} (u_{\ell,k} - w_{\text{idx}(\ell,k)})^2, \tag{2.27d}$$

$$\boldsymbol{u} \in \mathbb{R}^{|\textbf{leaves}(T)| \times 2^n}, \ \rho_{\text{trn}}, \rho_{\text{intra}}, \rho_{\text{inter}} \in \mathbb{R}, \ \boldsymbol{z} \in \mathbb{R}^{|\textbf{leaves}(T)|}, \ \boldsymbol{w} \in \mathbb{R}^{|W|}. \tag{2.27e}$$

Variable ρ_{trn} computes the mean squared training error. Variable ρ_{intra} sums up, over all hyper-rectangles, the squared deviations between the function values at the vertices with the average value z_ℓ. If $\rho_{\text{intra}} = 0$, the ML-tree is piecewise-constant. Variable ρ_{inter} sums up, over all vertices $t_i \in W$, the squared deviations between function values at the vertex in all hyper-rectangles that contain it and their average value w_i. If the tree structure is given, (2.27) is a convex quadratic program.

We compare ML-trees with standard decision trees. Given a data set, we train three (α, β)-ML-trees where (α, β) in (2.27) is chosen in $\{(\infty, 0), (0, \infty), (0.1, 0.1)\}$. The tree-structure for all the trees is the same and obtained using CART. The $(\infty, 0)$-ML-tree is the same as the standard decision tree.

We split the data-sets into training, validation, and testing data-sets. Each model is trained by choosing a depth from 2 to 7 that yields the smallest validation error. The data set consists of 15 nonlinear functions. The results, which can be found in Table 2.6, show that the testing error on $(0, \infty)$ and $(0.1, 0.1)$-ML-tree is smaller than that for the standard $(\infty, 0)$-ML-tree when the data is generated without/with Gaussian noise. In other words, piecewise-multilinear functions provide better fits as compared to piecewise-constant functions. Tree ensemble optimization with ML-trees leads to general multilinear optimization problems with continuous as well as discrete variables,

Table 2.6: Test errors of (α,β)-ML-tree models with $(\alpha,\beta) \in \{(\infty,0),(0,\infty),(0.1,0.1)\}$. Column $h(\boldsymbol{x})$ displays the nonlinear functions used in generating the data set. The second row displays parameters (α,β) used for training.

No.	$h(\boldsymbol{x})$	Without Gaussian noise			With Gaussian noise		
		$(\infty,0)$	$(0,\infty)$	$(0.1,0.1)$	$(\infty,0)$	$(0,\infty)$	$(0.1,0.1)$
1	$\prod_{i=1}^n x_i$	0.0024	**0.0000**	0.0004	0.0046	**0.0035**	0.0036
2	$\sum_{i=1}^n x_i^2$	0.0116	**0.0048**	0.0092	0.0194	**0.0147**	0.0160
3	$\sum_{i=1}^n x_i^3$	0.0137	**0.0058**	0.0110	0.0201	**0.0155**	0.0171
4	$\sqrt{\sum_{i=1}^n x_i^2}$	0.0050	**0.0021**	0.0037	0.0093	**0.0072**	0.0076
5	$\sqrt[3]{\sum_{i=1}^n x_i^3}$	0.0042	**0.0022**	0.0032	0.0076	**0.0060**	0.0062
6	$\sum_{i=1}^n \sqrt{x_i}$	0.0149	**0.0029**	0.0097	0.0208	0.0201	**0.0190**
7	$\sum_{i=1}^n \sqrt[3]{x_i}$	0.0193	**0.0060**	0.0137	0.0238	0.0231	**0.0223**
8	$\exp(\sum_{i=1}^n x_i/n)$	0.0042	**0.0002**	0.0025	0.0084	**0.0060**	0.0064
9	$\exp(\sum_{i=1}^n x_i^2/n)$	0.0046	**0.0018**	0.0036	0.0079	**0.0062**	0.0067
10	$\exp(\prod_{i=1}^n x_i)$	0.0024	**0.0013**	**0.0013**	0.0070	**0.0060**	0.0061
11	$\log(1+\sum_{i=1}^n x_i)$	0.0037	**0.0004**	0.0023	0.0077	**0.0060**	**0.0060**
12	$\log(1+\sum_{i=1}^n x_i^2)$	0.0047	**0.0021**	0.0036	0.0080	**0.0060**	0.0064
13	$\log(1+\prod_{i=1}^n x_i)$	0.0019	**0.0001**	0.0005	0.0033	**0.0025**	0.0026
14	$\frac{x_1^2+x_2^2+x_1-x_2+1}{(x_3-1.5)(x_4-1.5)}$	0.0518	**0.0124**	0.0214	0.0672	**0.0466**	0.0485
15	$\frac{\exp(x_1x_2x_3x_4)}{x_1^2+x_2^2-x_3x_4+3}$	0.0018	0.0011	**0.0010**	0.0018	0.0011	**0.0010**

for which our framework yields tight relaxations.

2.6.4 Supplements: Generation of Multi-Commodity Flow Problem Instances

In this section, we introduce the procedure we use to generate instances of the multi-commodity flow problem used in the computational experiments. First, we randomly generate supply and demand with the idea that there is 20% more supply than demand. We set the total demand (resp. total supply) of the i^{th} commodity to be $D_i = 100 \cdot 3^{i-1}$ (resp. $S_i = 1.2D_i$) for $i \in [C]$. For $i \in [C]$, $s_{u,i}$ for $u \in U$ represents the i^{th} commodity's supply and $d_{v,i}$ for $v \in V$ represents the i^{th} commodity's demand. We randomly distribute the total demand (resp. supply) to the demand (resp. supply) nodes using the following procedure: (i) pick $p_{v,i} = \text{Uniform}(1,2)$ for all $v \in V$ and $i \in [C]$ (resp. pick $p_{u,i} = \text{Uniform}(1,2)$ for all $u \in U$ and $i \in [C]$), (ii) compute

$d_{v,i} = \lfloor \frac{p_{v,i}}{\sum_{v \in V} p_{v,i}} D_i \rfloor$ for all $v \in V$ and $i \in [C]$ (resp. $s_{u,i} = \lceil \frac{p_{u,i}}{\sum_{u \in U} p_{u,i}} S_i \rceil$ for all $u \in U$ and $i \in [C]$). Second, we deduce some suitable values for capacities, that is, for $i \in [C]$ and $e \in E$, the arc capacity $c_{e,i}$ for the i^{th} commodity on arc e is computed as $\lceil 0.25\,(s_{u,i} + d_{v,i}) \rceil$. Third, for each arc $e \in E$, we define a hidden cost function $g_e(\boldsymbol{z_e}) = (1 + p_1(\boldsymbol{z_e}) + p_2(\boldsymbol{z_e}))\,||\boldsymbol{z_e}||_2$ where $c_e = \sum_{i \in [C]} c_{e,i}$, $p_1(\boldsymbol{z_e}) = \frac{\max\{0.25c_e - \sum_{i \in [C]} z_{e,i}, 0\}}{0.25c_e} \in [0,1]$, and $p_2(\boldsymbol{z_e}) = \frac{5\max\{\sum_{i \in [C]} z_{e,i} - 0.75c_e, 0\}}{0.25c_e} \in [0,5]$ for all $e \in E$. The underlying idea is that $g_e(\boldsymbol{z_e})$ is proportional to the Euclidean distance between the flow $\boldsymbol{z_e}$ and the zero flow, and that there are penalties when total flow on arc e is less than 25% or more than 75% of its total capacity c_e. We then generate the data points for the cost incurred on arc e as $\{(\bar{\boldsymbol{z}}, g_e(\bar{\boldsymbol{z}}))\}_{\bar{\boldsymbol{z}} \in \mathbb{Z} \cap \prod_{i \in [C]} [0, c_{e,i}]}$, that is, we compute the value of g at all the integer points in the hyper-rectangular domain bounded by individual capacities. We train a random forest model using `RandomForestRegressor` in `scikit-learn` package in Python (Pedregosa et al., 2011) with parameters (D,T) to describe $\{(\bar{\boldsymbol{z}}_{\boldsymbol{e}}^{\boldsymbol{j}}, g_e(\bar{\boldsymbol{z}}_{\boldsymbol{e}}^{\boldsymbol{j}}))\}_{j \in [n_{\text{data}}]}$ for each arc $e \in E$ and use those trees in the multi-commodity transportation problem instance.

2.6.5 Supplements: Formulations Used in the Experiments

In this section, we provide explicit formulations for (TEO), (TEOM), (MC), (DCC), and (DLog). For $e \in E$ and $i \in [C]$, we let $d_{e,i,1} \leq \ldots \leq d_{e,i,K_{e,i}}$ be the unique split values of the flow variables $z_{e,i}$ in the decision trees associated with the cost on arc e. Clearly, $K_{e,i}$ represents the number of unique split values on $z_{e,i}$. We introduce interval indicator variables (in incremental form) for $z_{e,i}$ that we call $x_{e,i,j}$, so that $x_{e,i,j} = \mathbb{1}[z_{e,i} \leq d_{e,i,j}]$. Let **leaves**$(e,t)$ (resp. **nleaves**(e,t)) be the set of leaves (resp. non-leaves) in the t^{th} tree describing the cost on arc $e \in E$. For $e \in E$, $t \in [T_e]$, $\ell \in \textbf{leaves}(e,t)$, we denote by $\delta_{e,t,\ell,i}$ the interval corresponding to the projection in the space of $z_{e,i}$ of the hyper-rectangle of leaf ℓ in the t^{th} tree associated with arc $e \in E$. To streamline presentation, we use $x_{e,i,0} := 0$, $x_{e,i,K_{e,i}+1} := 1$, $d_{e,i,0} := -\infty$, and $d_{e,i,K_{e,i}+1} := \infty$ for all $i \in [C]$ and $e \in E$. Also, $\ell \in \textbf{leaves}(e,t)$ for $e \in E$ and $t \in [T_e]$ indicates the index of the leaf. The explicit formulation of (TEO) is as follows:

$$\min \quad \sum_{e \in E} \sum_{t \in [T_e]} \sum_{\ell \in \textbf{leaves}(e,t)} p_{e,t,\ell}\, y_{e,t,\ell} \tag{2.28a}$$

$$\text{s.t.} \quad \sum_{e=(u',v')\in E: u'=u} z_{e,i} \le s_{u,i}, \qquad \forall u \in U, \forall i \in [C], \tag{2.28b}$$

$$\sum_{e=(u',v')\in E: v'=v} z_{e,i} \ge d_{v,i}, \qquad \forall v \in V, i \in [C], \tag{2.28c}$$

$$z_{e,i} \ge \sum_{j\in[K_{e,i}]} (d_{e,i,j-1} - d_{e,i,j}) x_{e,i,j}, \qquad \forall e \in E, \forall i \in [C], \tag{2.28d}$$

$$z_{e,i} \le c_{e,i} + \sum_{j\in[K_{e,i}]} (d_{e,i,j} - d_{e,i,j+1}) x_{e,i,j}, \qquad \forall e \in E, \forall i \in [C], \tag{2.28e}$$

$$\sum_{\ell\in\mathbf{leaves}(e,t):\delta_{e,t,\ell,i}\subseteq[d_{e,i,j_1},d_{e,i,j_2}]} y_{e,t,\ell} \le x_{i,j_2} - x_{i,j_1}, \quad \forall e \in E, \forall t \in [T_e], \forall i \in [C], \forall j_1, j_2 \in [0..(K_{e,i}+1)] : j_1 < j_2, \tag{2.28f}$$

$$x_{e,i,j} \le x_{e,i,j+1}, \qquad \forall e \in E, \forall i \in [C], \forall j \in [K_{e,i}-1], \tag{2.28g}$$

$$x_{e,i,j} \in \{0,1\}, \qquad \forall e \in E, \forall i \in [C], \forall j \in [K_{e,i}], \tag{2.28h}$$

$$\boldsymbol{y_{e,t}} \in \Delta^{|\mathbf{leaves}(e,t)|}, \qquad \forall e \in E, \forall t \in [T_e], \tag{2.28i}$$

$$z_{e,i} \in \mathbb{R}, \qquad \forall e \in E, \forall i \in [C]. \tag{2.28j}$$

We next obtain (TEOM) from (TEO). Let $\mathbf{lchild}(s)$ (resp. $\mathbf{rchild}(s)$) be the set of leaves under the left (resp. right) child of s. For $e \in E$, $t \in [T_e]$, $s \in \mathbf{nleaves}(e,t)$, let $i(s)$ (resp. $j(s)$) be the index of the independent variable (resp. the split value) associated with the query on n, that is, "$z_{e,i(s)} \le d_{e,i(s),j(s)}?''$ We obtain (TEOM) from (TEO) by replacing (2.28f) with (2.29):

$$\sum_{\ell\in\mathbf{lchild}(s)} y_{e,t,\ell} \le x_{e,i(s),j(s)}, \qquad \forall e \in E, \forall t \in [T_e], \forall s \in \mathbf{nleaves}(e,t), \tag{2.29a}$$

$$\sum_{\ell\in\mathbf{rchild}(s)} y_{e,t,\ell} \le 1 - x_{e,i(s),j(s)}, \qquad \forall e \in E, \forall t \in [T_e], \forall s \in \mathbf{nleaves}(e,t), \tag{2.29b}$$

that is,

$$\begin{aligned} \min \quad & \sum_{e\in E}\sum_{t\in[T_e]}\sum_{\ell\in\textbf{leaves}(e,t)} p_{e,t,\ell}y_{e,t,\ell} \\ \text{s.t.} \quad & \text{(2.28b)–(2.28e), (2.28g)–(2.28j), (2.29).} \end{aligned}$$

Formulations (MC), (DCC), and (DLog) are developed based on general disjunctive programming techniques (Vielma et al., 2010). The formulations use $\boldsymbol{y}$ as binary variable to indicate active hyper-rectangles. For $e \in E$, $t \in [T_e]$, $i \in [C]$, $\ell \in \textbf{leaves}(e,t)$, let $\text{lb}(e,t,i,\ell)$ (resp. $\text{ub}(e,t,i,\ell)$) be the lower- (resp. upper-) limit on $z_{e,i}$ of the hyper-rectangle corresponding to leaf ℓ. The explicit formulation of (MC) is as follows:

$$\min \quad \sum_{e\in E}\sum_{t\in[T_e]}\sum_{\ell\in\textbf{leaves}(e,t)} p_{e,t,\ell}y_{e,t,\ell} \tag{2.30a}$$

$$\text{s.t.} \quad \sum_{e=(u',v')\in E:u'=u} z_{e,i} \le s_{u,i}, \qquad \forall u \in U, \forall i \in [C], \tag{2.30b}$$

$$\sum_{e=(u',v')\in E:v'=v} z_{e,i} \ge d_{v,i}, \qquad \forall v \in V, \forall i \in [C], \tag{2.30c}$$

$$z_{e,i} = \sum_{\ell\in\textbf{leaves}(e,t)} z_{e,i,t,\ell}, \qquad \forall e \in E, \forall t \in [T_e], \tag{2.30d}$$

$$z_{e,i,t,\ell} \ge \max(0, \text{lb}(e,t,i,\ell))y_{e,t,\ell}, \qquad \forall e \in E, \forall i \in [C], \forall t \in [T_e], \forall \ell \in \textbf{leaves}(e,t), \tag{2.30e}$$

$$z_{e,i,t,\ell} \le \min(c_{e,i}, \text{ub}(e,t,i,\ell))y_{e,t,\ell}, \qquad \forall e \in E, \forall i \in [C], \forall t \in [T_e], \forall \ell \in \textbf{leaves}(e,t), \tag{2.30f}$$

$$\boldsymbol{y_{e,t}} \in \Delta_{0,1}^{|\textbf{leaves}(e,t)|}, \qquad \forall e \in E, \forall t \in [T_e], \tag{2.30g}$$

$$z_{e,i} \in \mathbb{R}, \qquad \forall e \in E, \forall i \in [C], \tag{2.30h}$$

$$z_{e,i,t,\ell} \in \mathbb{R}, \qquad \forall e \in E, \forall i \in [C], \forall t \in [T_e], \ell \in \textbf{leaves}(e,t). \tag{2.30i}$$

For $e \in E$, $t \in [T_e]$, $\ell \in \textbf{leaves}(e,t)$, let $Q_{e,t,\ell} \subseteq \prod_{i\in[C]}[0, c_{e,i}]$ be the hyper-rectangle corresponding to the leaf ℓ in tree t. Let $\bar{\boldsymbol{z}}_{e,t,\ell,k} \in \mathbb{R}^C$ be the k^{th} extreme point (among

2^C of them) of $Q_{e,t,\ell}$. The explicit formulation of (DCC) is as follows:

$$
\begin{aligned}
\min \quad & \sum_{e\in E}\sum_{t\in[T_e]}\sum_{\ell\in\mathbf{leaves}(e,t)} p_{e,t,\ell}y_{e,t,\ell} && && \text{(2.31a)}\\
\text{s.t.} \quad & \sum_{e=(u',v')\in E:u'=u} z_{e,i} \le s_{u,i}, && \forall u\in U, \forall i\in[C], && \text{(2.31b)}\\
& \sum_{e=(u',v')\in E:v'=v} z_{e,i} \ge d_{v,i}, && \forall v\in V, \forall i\in[C], && \text{(2.31c)}\\
& \boldsymbol{z_e} = \sum_{\ell\in\mathbf{leaves}(e,t)}\sum_{k\in[2^C]} \lambda_{e,t,\ell,k}\bar{\boldsymbol{z}}_{e,t,\ell,k}, && \forall e\in E, \forall t\in[T_e], && \text{(2.31d)}\\
& \sum_{\ell\in\mathbf{leaves}(e,t)}\sum_{k\in[2^C]} \lambda_{e,t,\ell,k} = 1, && \forall e\in E, \forall t\in[T_e], && \text{(2.31e)}\\
& \sum_{k\in[2^C]} \lambda_{e,t,\ell,k} \le y_{e,t,\ell}, && \forall e\in E, \forall t\in[T_e], \forall \ell\in\mathbf{leaves}(e,t), && \text{(2.31f)}\\
& \boldsymbol{y_{e,t}} \in \Delta_{0,1}^{|\mathbf{leaves}(e,t)|}, && \forall e\in E, \forall t\in[T_e], && \text{(2.31g)}\\
& z_{e,i}\in\mathbb{R}, && \forall e\in E, \forall i\in[C], && \text{(2.31h)}\\
& \lambda_{e,t,\ell,k}\in\mathbb{R}_+, && \forall e\in E, && \\
& && \forall t\in[T_e], \forall \ell\in\mathbf{leaves}(e,t), k\in[2^C]. && \text{(2.31i)}
\end{aligned}
$$

The explicit formulation of (DLog) is as follows:

$$
\begin{aligned}
\min \quad & \sum_{e\in E}\sum_{t\in[T_e]}\sum_{\ell\in\mathbf{leaves}(e,t)} p_{e,t,\ell}y_{e,t,\ell} && && \text{(2.32a)}\\
\text{s.t.} \quad & \sum_{e=(u',v')\in E:u'=u} z_{e,i} \le s_{u,i}, && \forall u\in U, \forall i\in[C], && \text{(2.32b)}\\
& \sum_{e=(u',v')\in E:v'=v} z_{e,i} \ge d_{v,i}, && \forall v\in V, \forall i\in[C], && \text{(2.32c)}\\
& \boldsymbol{z_e} = \sum_{\ell\in\mathbf{leaves}(e,t)}\sum_{k\in[2^C]} \lambda_{e,t,\ell,k}\bar{\boldsymbol{z}}_{e,t,\ell,k}, && \forall e\in E, \forall t\in[T_e], && \text{(2.32d)}\\
& \sum_{\ell\in\mathbf{leaves}(e,t)}\sum_{k\in[2^C]} \lambda_{e,t,\ell,k} = 1, && \forall e\in E, \forall t\in[T_e], && \text{(2.32e)}\\
& \sum_{\ell\in\mathbf{leaves}(e,t):\ell\%2^k<2^{k-1}}\ \sum_{k\in[2^C]} \lambda_{e,t,\ell,k} \le y_{e,t,\ell}, && \forall e\in E, \forall t\in[T_e], &&
\end{aligned}
$$

$$\forall k \in [\lceil \log_2 |\mathbf{leaves}(e,t)| \rceil], \tag{2.32f}$$

$$\sum_{\ell \in \mathbf{leaves}(e,t) : \ell \% 2^k \geq 2^{k-1}} \sum_{k \in [2^C]} \lambda_{e,t,\ell,k} \leq 1 - y_{e,t,\ell}, \quad \forall e \in E, \forall t \in [T_e],$$

$$\forall k \in [\lceil \log_2 |\mathbf{leaves}(e,t)| \rceil], \tag{2.32g}$$

$$\boldsymbol{y_{e,t}} \in \{0,1\}^{\lceil \log_2 |\mathbf{leaves}(e,t)| \rceil}, \quad \forall e \in E, \forall t \in [T_e], \tag{2.32h}$$

$$z_{e,i} \in \mathbb{R}, \quad \forall e \in E, \forall i \in [C], \tag{2.32i}$$

$$\lambda_{e,t,\ell,k} \in \mathbb{R}_+, \quad \forall e \in E, \forall t \in [T_e], \forall \ell \in \mathbf{leaves}(e,t), \forall k \in [2^C], \tag{2.32j}$$

where, for positive integers a and b, we use the notation $a\%b$ to denote the remainder of the division of a by b.

2.6.6 Supplements: Unique Multilinear Function Over a Hyper-Rectangle

Lemma 7. *Let $\{\bar{\boldsymbol{x}}^{\mathbf{1}}, \ldots, \bar{\boldsymbol{x}}^{\boldsymbol{m}}\}$ be the set of vertices of a full-dimensional (axis-parallel) hyper-rectangle $\mathcal{H} \subset \mathbb{R}^n$, i.e., $m = 2^n$. Given $\bar{\boldsymbol{y}} \in \mathbb{R}^m$, there exists a unique vector $\boldsymbol{c} \in \mathbb{R}^m$ such that the function $g(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$ satisfies $g(\bar{\boldsymbol{x}}^i) = \bar{y}_i$ for all $i \in [m]$.*

Proof. We prove the statement by induction on n. In the proof, we only consider unit hyper-cubes $[0,1]^n$ in $\mathbb{R}^n$ because the result easily generalizes to other (axis-parallel) full-dimensional hyper-rectangles. When $n = 1$, the result is clear. In this case, $\mathcal{H} = [0,1]$ with $\bar{x}^1 = 0$ and $\bar{x}^2 = 1$. Given $\bar{y}_1, \bar{y}_2 \in \mathbb{R}$, there exists a unique linear (multilinear reduces to linear when $n = 1$) function g that is such that $g(\bar{x}^1) = \bar{y}_1$ and $g(\bar{x}^2) = \bar{y}_2$, *i.e.*, $c_\emptyset = \bar{y}_1$ and $c_{\{1\}} = \bar{y}_2 - \bar{y}_1$.

Assume now that the statement holds for all hypercubes $\mathcal{H} \subseteq \mathbb{R}^n$ and for all vectors $\boldsymbol{y} \in \mathbb{R}^{2^n}$ when $n \leq k$ for some positive integer k. We will show that the statement holds for $n = k+1$. Let $\{\bar{\boldsymbol{x}}^1, \ldots, \bar{\boldsymbol{x}}^m\}$ be the set of vertices of $\mathcal{H} \subset \mathbb{R}^{k+1}$ where $m = 2^{k+1}$ and let $\bar{\boldsymbol{y}} \in \mathbb{R}^m$. We partition $[m]$ into $I_j = \{i \in [m] \mid \bar{x}^i_{k+1} = j\}$ for $j \in \{0,1\}$. For $j \in \{0,1\}$, there exists, by the inductive hypothesis, a unique vector $\boldsymbol{c}^j$ such that

$g_j := \sum_{S\subseteq[k]} c_S^j \prod_{i\in S} x_i$ takes values $\{\bar{y}_i\}_{i\in I_j}$ at the extreme points $\{(\bar{x}_1^i,\ldots,\bar{x}_k^i)\}_{i\in I_j}$ of the hyper-rectangle these points define in $\mathbb{R}^k$. It is then simple to verify that the function $g(x_1,\ldots,x_{k+1}) = (1-x_{k+1})g_0(x_1,\ldots,x_k) + x_{k+1}g_1(x_1,\ldots,x_k)$ is of the form $\sum_{S\subseteq[k+1]} \tilde{c}_S \prod_{i\in S} x_i$ for some suitable values of $\tilde{c}_S$ and takes the desired values at the extreme points of $\mathcal{H}$. This shows that at least one vector $\boldsymbol{c}$ with the desired property exists.

We next show that the vector $\boldsymbol{c}$ is unique. Take any function $g(x_1,\ldots,x_{k+1}) = \sum_{S\subseteq[k+1]} c_S \prod_{i\in S} x_i$ satisfying $g(\bar{\boldsymbol{x}}^i) = \bar{y}_i$ for all $i \in [m]$. By factoring out variable x_{k+1}, it can be rewritten as

$$g(x_1,\ldots,x_{k+1}) = \sum_{S\subseteq[k]} c_S \prod_{i\in S} x_i + \sum_{S\subseteq[k]} c_{S\cup\{k+1\}} x_{k+1} \prod_{i\in S} x_i.$$

Plugging the values of 0 or 1 for x_{k+1} in this expression yields

$$\begin{aligned} g(x_1,\ldots,x_k,0) &= \sum_{S\subseteq[k]} c_S \prod_{i\in S} x_i, \\ g(x_1,\ldots,x_k,1) &= \sum_{S\subseteq[k]} (c_S + c_{S\cup\{k+1\}}) \prod_{i\in S} x_i. \end{aligned}$$

For $j \in \{0,1\}$, $g(x_1,\ldots,x_k,j)$ is of the form $\sum_{S\subseteq[k]} \tilde{c}_S^j \prod_{i\in S} x_i$ for suitable $\tilde{c}_S^j$ and takes the values $\{\bar{y}_i\}_{i\in I_j}$ at the vertices $\{(\bar{x}_1^i,\ldots,\bar{x}_k^i)\}_{i\in I_j}$ of a full-dimensional hypercube in $\mathbb{R}^k$. By induction, the coefficient $\{c_S\}_{S\subseteq[k]}$ and $\{c_S + c_{S\cup\{k+1\}}\}_{S\subseteq[k]}$ are uniquely determined. This implies that coefficients $\{c_S\}_{S\subseteq[k+1]}$ are also uniquely determined. This completes the proof of the inductive step. □

2.7 Conclusion

This chapter investigates and exposes new connections between multilinear optimization problems and tree ensemble optimization problems. In particular, we show that multilinear optimization problems involving discrete variables are polynomially reducible to TEO problems and vice-versa. We then use insights from convexification of multilinear functions to improve relaxations for tree ensemble optimization problems and uses tree ensemble representations to develop new convex relaxations for multilinear polytopes. We perform computational experiments to show that our formulations for TEO are tighter and have distinct computational advantages over existing formulations based on the modeling of piecewise-linear functions and/or existing tree ensemble formulations. Finally, we conclude by proposing a training model similar to decision-trees but inspired by the observation that TEO models are less expressive than multilinear problems involving continuous variables. We show that our proposed multilinear-decision-tree model fits nonlinear functions better and reduces testing error significantly. The use of such generalized decision trees in optimization models would further couple multilinear optimization and TEO.

Chapter 3

Piecewise Polyhedral Relaxations of Multilinear Optimization

The work described in this chapter was performed under the guidance of Prof. Jean-Philippe P. Richard and Prof. Mohit Tawarmalani. The research presented in this chapter was supported by grants 1727989 and 1917323 from the National Science Foundation Division of Civil, Mechanical and Manufacturing Innovation.

3.1 Preface

In this chapter, we consider piecewise polyhedral relaxations (PPRs) of multilinear optimization problems over axis-parallel hyper-rectangular partitions of their domain. We improve formulations for PPRs by linking components that are commonly modeled independently in the literature. Numerical experiments with ALPINE, an open-source software for global optimization that relies on piecewise approximations of functions, show that the resulting formulations significantly speed-up the solver when compared to its default settings. Most results on piecewise functions in the literature assume that the partition is *regular*. Regular partitions arise when the domain of each individual input variable is divided into nonoverlapping intervals and when the partition of the overall domain is composed of all Cartesian products of these intervals. We provide the first polynomially-sized locally ideal formulation for general hyper-rectangular partitions. We also perform experiments that show that a formulation over non-regular partitions outperforms that over regular ones.

3.2 Introduction

We consider multilinear optimization problems of the form

$$
\begin{aligned}
\max \quad & \boldsymbol{c}_{\boldsymbol{z}}^{\intercal}\boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal}\boldsymbol{w} && && \text{(3.1a)}\\
\text{s.t.} \quad & A_{\boldsymbol{z}}\boldsymbol{z} + A_{\boldsymbol{w}}\boldsymbol{w} \le \boldsymbol{b}, && && \text{(3.1b)}\\
& w_j = f_j(\boldsymbol{z}_{I_j}), && \forall j \in [m], && \text{(3.1c)}\\
& \ell_i \le z_i \le u_i, && \forall i \in [n], && \text{(3.1d)}
\end{aligned}
$$

where m, n, $n_A \in \mathbb{Z}_+$, $\boldsymbol{c}_{\boldsymbol{z}} \in \mathbb{R}^n$, $\boldsymbol{c}_{\boldsymbol{w}} \in \mathbb{R}^m$, $A_{\boldsymbol{z}} \in \mathbb{R}^{n_A \times n}$, $A_{\boldsymbol{w}} \in \mathbb{R}^{n_A \times m}$, $\boldsymbol{b} \in \mathbb{R}^{n_A}$, $\boldsymbol{\ell}$ and $\boldsymbol{u} \in \mathbb{R}^n$, $f_j : \mathbb{R}^{|I_j|} \mapsto \mathbb{R}$ is a multilinear function of variables z_k with indices k in $I_j \subseteq [n]$ for all $j \in [m]$, $[a] := \{1, \dots, a\}$ for positive integer a, and where we use bold lowercase letters to denote vectors. Motivated by advances in integer programming solvers, there has been an interest in constructing discrete relaxations for mixed-integer nonlinear programming (MINLP) problems. A strategy adopted by MINLP solvers such as ALPINE (Nagarajan et al., 2019, 2016; Sundar et al., 2021b) and ANTIGONE (Misener and Floudas, 2014) is to introduce new variables for univariate functions and then use discretization strategies to relax (3.1). In this paper, we develop insights into this latter relaxation. We refer to subsets of $\mathbb{R}^n$ defined by constraints of the form (3.1d), with $\boldsymbol{\ell} < \boldsymbol{u}$, as *hyper-rectangles.* We denote the hyper-rectangle with lower bounds $\boldsymbol{\ell}$ and upper bounds $\boldsymbol{u}$ as $\mathcal{Z}(\boldsymbol{\ell}, \boldsymbol{u})$, which we abbreviate as $\mathcal{Z}$ when parameters $\boldsymbol{\ell}$ and $\boldsymbol{u}$ are clear from the context. For $I \subseteq [n]$, we denote by $\mathcal{Z}_I$ the hyper-rectangle $\mathcal{Z}(\boldsymbol{\ell}_I, \boldsymbol{u}_I)$ obtained by projecting $\mathcal{Z}$ over the space of $\boldsymbol{z}_I$ variables.

We investigate mixed integer programming (MIP) models for the type of piecewise polyhedral relaxations of (3.1) over hyper-rectangular partitions that we describe next. A collection $\{Q_i\}_{i \in [L]}$ of full-dimensional subsets of a compact set $S \subseteq \mathbb{R}^n$ is a *partition* of S if (i) $\bigcup_{i \in [L]} Q_i = S$ and (ii) the interiors of $\{Q_i\}_{i \in [L]}$ are pairwise disjoint. A partition $\{Q_i\}_{i \in [L]}$ of S is said to be *polyhedral* if Q_i is a polyhedron for all $i \in [L]$. A polyhedral partition $\{Q_i\}_{i \in [L]}$ of S is said to be *hyper-rectangular* (or equivalently that $\{Q_i\}_{i \in [L]}$ is a *hyper-rectangular partition (HP)*) if Q_i is a hyper-rectangle for all $i \in [L]$.

Let $h(\boldsymbol{z}) : \mathbb{R}^n \mapsto \mathbb{R}$ be a function whose domain is $\mathcal{Z}(\boldsymbol{\ell}, \boldsymbol{u})$ for some $\boldsymbol{\ell}, \boldsymbol{u} \in \mathbb{R}^n$ and let $\{Q_i\}_{i\in[L]}$ be a polyhedral partition of $\mathcal{Z}$. We denote by $\operatorname{vert} S$ the set of the extreme points of polytope S. A *piecewise polyhedral relaxation (PPR)* of the graph of h over $\{Q_i\}_{i\in[L]}$ is defined as a set $S = \bigcup_{i\in[L]} \bar{Q}_i$ where $\bar{Q}_i$ is a polytope in $\mathbb{R}^{n+1}$, $\operatorname{vert} Q_i = \operatorname{proj}_{\boldsymbol{z}} \operatorname{vert} \bar{Q}_i$, and $\{(\boldsymbol{z}, w) \in Q_i \times \mathbb{R} \mid w = h(\boldsymbol{z})\} \subseteq \bar{Q}_i$ for all $i \in [L]$. In general, there may not exist a PPR that is contained in all PPR that can be constructed over the same polyhedral partition of the domain. However, if the convex hull of the graph of $h(\boldsymbol{z})$ over $Q_{j,i}$ depends only on function value at the vertices of $Q_{j,i}$, as is the case when $h(\boldsymbol{z})$ is multilinear and the partition is hyper-rectangular, the smallest PPR (SPPR) can be constructed by choosing $\bar{Q}_i = \operatorname{conv}\{(\boldsymbol{z}, h(\boldsymbol{z}))\}_{\boldsymbol{z}\in\operatorname{vert} Q_i}$ for all $i \in [L]$. When $L = 1$, the SPPR of a multilinear function h describes the convex hull of the graph of h over its hyper-rectangular domain. Given a relaxation of a nonlinear function defined over a polyhedral partition, a tighter relaxation can typically be obtained by subdividing the original partition. We refer to a relaxation of an optimization problem Σ obtained by relaxing nonlinear or nonconvex functions using PPRs as a *PPR of* Σ.

In this chapter, we consider the PPR of (3.1) obtained by individually modeling the SPPR of each f_j over the HP $\{Q_{j,i}\}_{i\in[L_j]}$ of $\mathcal{Z}_{I_j}$ for $j \in [m]$, since it is commonly used in software implementations:

$$\begin{aligned}
\max \quad & \boldsymbol{c}_{\boldsymbol{z}}^\intercal \boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^\intercal \boldsymbol{w} && \text{(3.2a)}\\
\text{s.t.} \quad & A_{\boldsymbol{z}} \boldsymbol{z} + A_{\boldsymbol{w}} \boldsymbol{w} \le \boldsymbol{b}, && \text{(3.2b)}\\
& \begin{pmatrix} \boldsymbol{z}_{I_j} \\ w_j \end{pmatrix} \in \bigcup_{i\in[L_j]} \operatorname{conv}\left\{ \begin{pmatrix} \bar{\boldsymbol{z}} \\ f_j(\bar{\boldsymbol{z}}) \end{pmatrix} \right\}_{\bar{\boldsymbol{z}}\in Q_{j,i}}, \qquad \forall j \in [m]. && \text{(3.2c)}
\end{aligned}$$

We refer to (3.2) as an individual piecewise polyhedral relaxation (IPPR) of (3.1). In (3.2), we relaxed each function f_j independently.

Throughout this chapter, we use a regular lowercase symbol to represent a variable (such as $\boldsymbol{z}$) and use a bar or a hat above the symbol (such as $\bar{\boldsymbol{z}}$ and $\hat{\boldsymbol{z}}$) to represent the variable at a certain point. Individual relaxation is a common technique to obtain relaxations of nonlinear optimization problems. McCormick's relaxation (McCormick, 1976) is one such example that is commonly used. In this chapter, we seek to derive constraints that improve individual relaxation-based formulations by taking advantage of the connections they share. For example, if there exists i, i' such that, it is known that $Q_{j,i} = Q_{j',i'}$ (or, more generally if $Q_{j,i}$ when projected to $z_{I_{j'}}$ yields $Q_{j',i'}$) then we will construct a tighter relaxation by simultaneously convexifying f_j and $f_{j'}$ over $Q_{j,i}$. In doing so, our formulations do not require additional variables.

A HP $\{Q_i\}_{i\in[L]}$ of a hyper-rectangle $S \subseteq \mathbb{R}^n$ is said to be *regular* (or equivalently that $\{Q_i\}_{i\in[L]}$ is a *regular hyper-rectangular partition (RHP)*) if, for all coordinate axes $v \in [n]$, there exists a collection of intervals $\delta_{v,k} := [d_{v,k}, d_{v,k+1}]$ for $k \in [D_v - 1]$, where $d_{v,1}, \dots, d_{v,D_v} \in \mathbb{R}$ are sorted in increasing order, such that each element Q_i is of the form $\prod_{v\in[n]} \delta_{v,k_{v,i}}$ for some $k_{v,i} \in [D_v - 1]$ for all $v \in [n]$. Formulations of piecewise approximations in the literature (Vielma et al., 2010; Huchette and Vielma, 2022; Nagarajan et al., 2019; Sundar et al., 2021a) often assume that the partition is regular.

The theoretical contributions of this chapter include (i) valid inequalities for (3.1) that tighten formulations for (3.2), and (ii) the first locally ideal formulations for (3.2) over (non-regular) HPs. The tightening exploits the shared variables across multilinear terms by interpreting convex multipliers of vertices of $Q_{j,i}$ as multilinear expressions. This interpretation also makes it possible to use recent developments on relaxations for multilinear optimization problems in the construction of the locally-ideal formulations mentioned above . We modify ALPINE, an open-source global solver for MINLPs, to use these formulations and show that this substantially improves computational performance.

3.3 Mixed-Integer Linear Formulations Over RHPs

In this section, we consider (3.2) over RHPs. We assume that partitions $\{Q_{j,i}\}_{i\in[L_j]}$ for all $j \in [m]$ share common discretization points on their axes. This assumption is prevalent in the literature; see (Vielma et al., 2010; Sundar et al., 2021a).

We provide a new mixed-integer linear programming (MILP) formulation for (3.2) that we prove is locally ideal and does not require too many variables. Moreover, we develop a mixed-integer multilinear programming formulation denoted by (MLP) for (3.1) and, via its linearization, further tighten the proposed MILP formulation. Finally, we perform computational experiments by integrating some of our results into ALPINE, since this code uses PPRs to obtain bounds when solving MINLPs to global optimality.

3.3.1 A Locally Ideal MILP Formulation for PPR Over RHPs

We first describe an MILP formulation for (3.2), which we refer to as (IPPR1). We use five types of decision variables: $\boldsymbol{z}$, $\boldsymbol{w}$, $\boldsymbol{\lambda}$, $\boldsymbol{\rho}$, and $\boldsymbol{x}$. Variables $\boldsymbol{z}$ and $\boldsymbol{w}$ are the same variables used in (3.1). Binary variable $x_{v,j}$ indicates an interval $\delta_{v,j}$ that contains z_v for all $v \in [n]$ and for all $j \in [D_v - 1]$. Variable $\boldsymbol{\lambda}$ represents the convex combination weights used to express the values of $\boldsymbol{w}$. For $I \subseteq [n]$, we denote by $\bar{\mathcal{Z}}_I := \prod_{v \in I} \{d_{v,k}\}_{k \in [D_v]}$ the set of all vertices that can be used in convex combinations in the space of $\boldsymbol{z}_I$. Variable $\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}$ indicates the convex combination weight for vertex $\bar{\boldsymbol{z}}$ in the space of $\boldsymbol{z}_I$ for all $I \in \mathcal{I}$ and for all $\bar{\boldsymbol{z}} \in \mathcal{Z}_I$ where $\mathcal{I} := \{I \subseteq [n] \mid \exists j \in [m] : I = I_j\}$ is the collection of all nonduplicate sets I_j. Variable $\rho_{v,k,\text{lb}}$ (resp. $\rho_{v,k,\text{ub}}$) represents the accumulated convex combination weight on the lower-bound (resp. upper-bound) of interval $\delta_{v,k}$ on the z_v-axis when $x_{v,k} = 1$ for all $v \in [n]$ and for all $k \in [D_v - 1]$. For $I \subseteq [n]$, $\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I$, and $v \in I$, we denote by $\bar{\boldsymbol{z}}_{(v)}$ the component corresponding to z_v in $\bar{\boldsymbol{z}}$. For a positive integer K, we use $\Delta^K := \{\boldsymbol{x} \in \mathbb{R}_+^K \mid \sum_{j \in [K]} x_j = 1\}$ to denote the simplex having as vertices the K principal vectors of $\mathbb{R}^K$ and use $\Delta_{0,1}^K$ to denote its vertices. Finally, we use the convention that $\rho_{v,0,\text{ub}} = \rho_{v,D_v,\text{lb}} = 0$ for all $I \in \mathcal{I}$ and for all $v \in I$. (IPPR1) can then be described as follows:

$$\max \quad \boldsymbol{c}_{\boldsymbol{z}}^{\intercal} \boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal} \boldsymbol{w} \tag{3.3a}$$

$$\text{s.t.} \quad A_{\boldsymbol{z}} \boldsymbol{z} + A_{\boldsymbol{w}} \boldsymbol{w} \leq \boldsymbol{b}, \tag{3.3b}$$

$$z_v = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I} \bar{\boldsymbol{z}}_{(v)} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}, \qquad \forall I \in \mathcal{I}, \forall v \in I, \tag{3.3c}$$

$$w_j = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{I_j}} f_j(\bar{\boldsymbol{z}}) \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{I_j}}, \qquad \forall j \in [m], \tag{3.3d}$$

$$\sum_{\bar{z} \in \bar{\mathcal{Z}}_I : \bar{z}_{(v)} = d_{v,k}} \lambda_{\bar{z}}^{\boldsymbol{z}_I} = \rho_{v,k-1,\text{ub}} + \rho_{v,k,\text{lb}} \qquad \forall I \in \mathcal{I}, \forall v \in I, \forall k \in [D_v], \tag{3.3e}$$

$$\rho_{v,k,\text{lb}} + \rho_{v,k,\text{ub}} \leq x_{v,k} \qquad \forall v \in [n], \forall k \in [D_v - 1], \tag{3.3f}$$

$$\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I} \geq 0, \qquad \forall I \in \mathcal{I}, \forall \bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I, \tag{3.3g}$$

$$\boldsymbol{\rho_v} = \{\rho_{v,k,a}\}_{k \in [D_v - 1], a \in \{\text{lb}, \text{ub}\}} \in \Delta^{2D_v - 2}, \qquad \forall v \in [n], \tag{3.3h}$$

$$\boldsymbol{x_v} \in \Delta_{0,1}^{D_v-1}, \qquad \forall v \in [n]. \tag{3.3i}$$

(IPPR1) models (3.2), *i.e.*, its projection in the space of $(\boldsymbol{z}, \boldsymbol{w})$ is equal to (3.2). We precisely show in Proposition 7 that (IPPR1) is a relaxation of (3.1). To this end, we first introduce a mixed-integer multilinear formulation for (3.1), which we denote by (MLP), that uses the same variables as in (3.3) except $\boldsymbol{\lambda}$. Then, we show that (IPPR1) can be obtained by linearizing (MLP). This, in turn, proves that (IPPR1) is a relaxation of (3.1). (MLP) is described as follows:

$$\max \quad \boldsymbol{c}_{\boldsymbol{z}}^{\intercal}\boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal}\boldsymbol{w} \tag{3.4a}$$
$$\text{s.t.} \quad A_{\boldsymbol{z}}\boldsymbol{z} + A_{\boldsymbol{w}}\boldsymbol{w} \le \boldsymbol{b}, \tag{3.4b}$$
$$z_v = \sum_{k\in[D_v]} d_{v,k}(\rho_{v,k-1,\text{ub}} + \rho_{v,k,\text{lb}}), \qquad \forall v \in [n], \tag{3.4c}$$
$$w_j = f_j(\boldsymbol{z}_{I_j}), \qquad \forall j \in [m], \tag{3.4d}$$
$$\rho_{v,k,\text{lb}} + \rho_{v,k,\text{ub}} \le x_{v,k} \qquad \forall v \in [n], \forall k \in [D_v - 1], \tag{3.4e}$$
$$\boldsymbol{\rho_v} = \{\rho_{v,k,a}\}_{k\in[D_v-1],a\in\{\text{lb},\text{ub}\}} \in \Delta^{2D_v-2}, \qquad \forall v \in [n], \tag{3.4f}$$
$$\boldsymbol{x_v} \in \Delta_{0,1}^{D_v-1}, \qquad \forall v \in [n]. \tag{3.4g}$$

The projection in the space of $(\boldsymbol{z}, \boldsymbol{w})$ variables of the feasible set of (MLP) is exactly the same as the feasible set of (3.1). This is because projecting the subsystem of (3.4c) and (3.4e)–(3.4g) onto $\boldsymbol{z}$ yields $\mathcal{Z}$. We show in Proposition 7 that any feasible solution of (MLP) can be mapped to a feasible solution of (IPPR1) that has the same values for $\boldsymbol{z}$ and $\boldsymbol{w}$, which proves that (IPPR1) is a relaxation of (3.1).

Proposition 7. *Given a feasible solution $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{w}}, \hat{\boldsymbol{\rho}}, \hat{\boldsymbol{x}})$ of (MLP), define*

$$\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I} := \prod_{v\in I} \hat{\lambda}_{\bar{\boldsymbol{z}}_{(v)}}^{z_v}, \qquad \forall I \in \mathcal{I},\ \forall \bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I, \tag{3.5}$$

where $\hat{\lambda}_{d_{v,k}}^{z_v} := \hat{\rho}_{v,k-1,ub} + \hat{\rho}_{v,k,lb}$ for all $v \in [n]$ and for all $k \in [D_v]$. Then, $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{w}}, \{\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}\}_{I\in\mathcal{I},\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_I}, \hat{\boldsymbol{\rho}}, \hat{\boldsymbol{x}})$ is feasible to (IPPR1).

Proof. It is sufficient to verify that the constraints containing $\{\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}\}_{I\in\mathcal{I},\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_I}$ variables, *i.e.*, (3.3c)–(3.3e) and (3.3g), are satisfied. By construction, $\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I} \ge 0$ for all $I \in \mathcal{I}$ and for all $\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I$, *i.e.*, (3.3g) is satisfied.

We next show that the constructed solution for (IPPR1) satisfies (3.3c)–(3.3e). To do so, we provide an equality relation (3.6) which converts an affine expression of $\{\hat{\lambda}^{\boldsymbol{z}_I}_{\bar{\boldsymbol{z}}}\}_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I}$ to a multlinear expression of $\{\hat{\lambda}^{z_v}_{d_{v,k}}\}_{v \in I, k \in [D_v]}$ for all $I \in \mathcal{I}$ and for all $v \in I$. Consider $I \in \mathcal{I}$ and, without loss of generality (WLOG), assume that $I = [|I|]$. Given a collection of functions $\{g_v(z_v)\}_{v \in I}$ where $g_v(z_v) : [d_{v,1}, d_{v,D_v}] \mapsto \mathbb{R}$ is piecewise-linear in z_v with breakpoints at $\{d_{v,k}\}_{k \in [D_v]}$, the following relation holds:

$$\begin{aligned}\sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I} \left(\prod_{v \in I} g_v(\bar{\boldsymbol{z}}_{(v)}) \right) \hat{\lambda}^{\boldsymbol{z}_I}_{\bar{\boldsymbol{z}}} &= \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I} \prod_{v \in I} g_v(\bar{\boldsymbol{z}}_{(v)}) \hat{\lambda}^{z_v}_{\bar{\boldsymbol{z}}_{(v)}} \\ &= \sum_{k_1 \in [D_1]} \cdots \sum_{k_s \in [D_{|I|}]} \prod_{v \in I} g_v(d_{v,k_v}) \hat{\lambda}^{z_v}_{d_{v,k_v}} = \prod_{v \in I} \sum_{k \in [D_v]} g_v(d_{v,k}) \hat{\lambda}^{z_v}_{d_{v,k}}, \qquad (3.6)\end{aligned}$$

where the first equality holds by definition, the second is obtained by expanding $\bar{\mathcal{Z}}_I$ into its elements, and the last is obtained by factoring out the terms not under the control of each sum.

We show that (3.3e) is satisfied for $I \in \mathcal{I}$, $v' \in I$, and $k' \in [D_v]$ using (3.6) by defining, for all $v \in I$ and for all $k \in [D_v]$, $g_v(d_{v,k}) = 1$ if $v \neq v'$ or $k = k'$ and $g_v(d_{v,k}) = 0$ otherwise. We write

$$\begin{aligned}\sum_{\bar{z} \in \bar{\mathcal{Z}}_I : \bar{z}_{(v')} = d_{v',k'}} \hat{\lambda}^{\boldsymbol{z}_I}_{\bar{z}} &= \sum_{\bar{z} \in \bar{\mathcal{Z}}_I} \left(\prod_{v \in I} g_v(\bar{\boldsymbol{z}}_{(v)}) \right) \hat{\lambda}^{\boldsymbol{z}_I}_{\bar{z}} = \prod_{v \in I} \sum_{k \in [D_v]} g_v(d_{v,k}) \hat{\lambda}^{z_v}_{d_{v,k}} \\ &= \sum_{k \in [D_{v'}]} g_{v'}(d_{v',k}) \hat{\lambda}^{z_{v'}}_{d_{v',k}} = \hat{\lambda}^{z_{v'}}_{d_{v',k'}} = \hat{\rho}_{v',k'-1,\mathrm{ub}} + \hat{\rho}_{v',k',\mathrm{lb}}, \qquad (3.7)\end{aligned}$$

where the first equality holds by the definition of $\{g_v(z_v)\}_{v \in I}$, the second holds by (3.6), the third holds because $\sum_{k \in [D_v]} g_v(d_{v,k}) \hat{\lambda}^{z_v}_{d_{v,k}} = \sum_{k \in [D_v]} \hat{\lambda}^{z_v}_{d_{v,k}} = 1$ for $v \neq v'$, the fourth holds by the definition of $g_{v'}(z_{v'})$, and the last holds by definition.

We next show that (3.3c) is satisfied. Consider $I \in \mathcal{I}$ and $S \subseteq I$. Define, for all $v \in I$ and for all $k \in [D_v]$, $g_v(d_{v,k}) = d_{v,k}$ if $v \in S$ and $g_v(d_{v,k}) = 1$ otherwise. Then, the following relation holds:

$$\sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_I}\left(\prod_{v\in S}\bar{\boldsymbol{z}}_{(v)}\right)\hat{\lambda}_{\bar{z}}^{\boldsymbol{z}_I}=\sum_{\bar{z}\in\bar{\mathcal{Z}}_I}\left(\prod_{v\in I}g_v(\bar{\boldsymbol{z}}_{(v)})\right)\hat{\lambda}_{\bar{z}}^{\boldsymbol{z}_I}=\prod_{v\in I}\sum_{k\in[D_v]}g_v(d_{v,k})\hat{\lambda}_{d_{v,k}}^{z_v}$$
$$=\prod_{v\in S}\sum_{k\in[D_v]}d_{v,k}\hat{\lambda}_{d_{v,k}}^{z_v}=\prod_{v\in S}\sum_{k\in[D_v]}d_{v,k}(\hat{\rho}_{v,k-1,\text{ub}}+\hat{\rho}_{v,k,\text{lb}})=\prod_{v\in S}\hat{z}_v,\quad (3.8)$$

where the first steps follow closely those of (3.7) and the last step holds by (3.4c). Applying (3.8) for all $I\in\mathcal{I}$ and for all $S=\{v\}\subseteq I$, we show that (3.3c) is satisfied for all $I\in\mathcal{I}$ and for all $v\in I$.

Finally, we show that (3.3d) is satisfied for $j\in[m]$. Multilinear function $f_j(z_{I_j})$ can be written as $\sum_{S\subseteq I_j}\alpha_S\prod_{v\in S}z_v$ for suitable coefficients $\alpha_S\in\mathbb{R}$ for all $S\subseteq I_j$. Then,

$$\sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_{I_j}}f_j(\bar{\boldsymbol{z}})\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{I_j}}=\sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_{I_j}}\left(\sum_{S\subseteq I_j}\alpha_S\prod_{v\in S}\bar{z}_v\right)\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{I_j}}$$
$$=\sum_{S\subseteq I_j}\alpha_S\left(\sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_{I_j}}\left(\prod_{v\in S}\bar{z}_v\right)\hat{\lambda}_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{I_j}}\right)=\sum_{S\subseteq I_j}\alpha_S\prod_{v\in S}\hat{z}_v=f_j(\hat{\boldsymbol{z}}_{I_j})=\hat{w}_j,$$

where the first and fourth equalities are obtained by using the expression of $f_j(\boldsymbol{z}_{I_j})$, the second is obtained by switching the order of summations, the third is obtained by (3.8), and the last holds by (3.4d). □

We next show that (IPPR1) is locally ideal and does not require too many variables. A formulation for (3.2) is said to be *locally ideal* if it is ideal when $m=1$. We say that a formulation for (3.2) is *polynomially-sized* if the total number of variables and constraints is polynomial in the total number of variables and constraints in (3.1) and in the total number of vertices used in convex combination expressions, *i.e.*, $\sum_{I\in\mathcal{I}}|\bar{\mathcal{Z}}_I|$.

Theorem 7. *(IPPR1) is a locally ideal polynomially-sized formulation for (3.2).*

Proof. It is clear that (IPPR1) is polynomially-sized. We show that (IPPR1) is locally ideal. To this end, we first temporarily introduce variables $\lambda_{d_{v,k}}^{z_v}$ to represent $\rho_{v,k-1,\text{ub}}+\rho_{v,k,\text{lb}}$ for all $v\in[n]$ and for all $k\in[D_v]$. Specifically, we construct a reformulation of (IPPR1) denoted by (IPPR1*) by introducing $\{\lambda_{d_{v,k}}^{z_v}\}_{v\in[n],k\in[D_v]}$, adding constraint $\lambda_{d_{v,k}}^{z_v}=\rho_{v,k-1,\text{ub}}+\rho_{v,k,\text{lb}}$ for all $v\in[n]$ and for all $k\in[D_v]$, and substituting $\lambda_{d_{v,k}}^{z_v}$

for $\rho_{v,k-1,\text{ub}} + \rho_{v,k,\text{lb}}$ in (3.3e) for all $v \in [n]$ and for all $k \in [D_v - 1]$. We further add constraints $\boldsymbol{\lambda}^{z_v} \in \Delta^{D_v}$ for all $v \in [n]$ which are redundant by (3.3h) for the purpose of the proof. It is clear that (IPPR1) is locally ideal if and only if (IPPR1*) is locally ideal.

Consider (IPPR1*) with $m = 1$. It follows that $\mathcal{I} = \{I_1\}$. We denote by S the feasible set of (IPPR1*). We show that the LP relaxation of S is $\text{conv}(S)$. We construct set S_0 by selecting all the constraints of (IPPR1*) only containing variables $(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}^{\boldsymbol{z}_{I_1}}, \{\boldsymbol{\lambda}^{z_v}\}_{v\in[n]})$ where $\boldsymbol{\lambda}^{\boldsymbol{z}_{I_1}} = \{\lambda^{\boldsymbol{z}_{I_1}}_{\bar{\boldsymbol{z}}}\}_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_{I_1}}$ and $\boldsymbol{\lambda}^{z_v} = \{\lambda^{z_v}_{d_{v,k}}\}_{k\in[D_v]}$ for all $v \in [n]$. For all $v \in [n]$, we construct set S_v by selecting all the constraints of (IPPR1*) only containing variables $(\boldsymbol{\lambda}^{z_v}, \boldsymbol{\rho}_v, \boldsymbol{x}_{\boldsymbol{v}})$. Then, we can reformulate S as

$$S = \left\{ X_{(\text{IPPR1*})} \;\middle|\; \begin{array}{ll} (\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}^{\boldsymbol{z}_{I_1}}, \{\boldsymbol{\lambda}^{z_v}\}_{v\in[n]}) \in S_0, & \\ (\boldsymbol{\lambda}^{z_v}, \boldsymbol{\rho}_v, \boldsymbol{x}_v) \in S_v, & \forall v \in [n] \end{array} \right\}.$$

where $X_{(\text{IPPR1*})} = (\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}^{\boldsymbol{z}_{I_1}}, \{\boldsymbol{\lambda}^{z_v}\}_{v\in[n]}, \boldsymbol{\rho}, \boldsymbol{x})$. Observe that S_0 and S_v share $\boldsymbol{\lambda}^{z_v}$ for all $v \in [n]$ and $S_{v'}$ and $S_{v''}$ do not share any variable for $v' \neq v'' \in [n]$. Moreover, $\boldsymbol{\lambda}^{z_v}$ forms a simplex for all $v \in [n]$. It follows that $\text{conv}(S)$ can be obtained by separately convexifying S_i for $i \in \{0, \dots, n\}$; see Lemma 6 in Chapter 2. We obtain that

$$\text{conv}(S) = \left\{ X_{(\text{IPPR1*})} \;\middle|\; \begin{array}{ll} (\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}^{\boldsymbol{z}_{I_1}}, \boldsymbol{\rho}, \{\boldsymbol{\lambda}^{z_v}\}_{v\in[n]}) \in \text{conv}(S_0), & \\ (\boldsymbol{\lambda}^{z_v}, \boldsymbol{x}_v) \in \text{conv}(S_v), & \forall v \in [n] \end{array} \right\}. \tag{3.9}$$

The constraints that belong to S_0 describe $\text{conv}(S_0)$ because there is no integral requirement in S_0, *i.e.*, $S_0 = \text{conv}(S_0)$. We next show that the LP relaxation of S_v is $\text{conv}(S_v)$ for all $v \in [n]$. Consider any fixed $v \in [n]$. The system in the space of $(\boldsymbol{\rho}_{\boldsymbol{v}}, \boldsymbol{x}_{\boldsymbol{v}})$ of (3.3f), (3.3h), and (3.3i) is referred to as the disaggregated convex combination formulation in (Vielma et al., 2010) and this formulation is known to be ideal. Further, the value of $\boldsymbol{\lambda}^{z_v}$ is dependent on the value of $\boldsymbol{\rho}_v$. It follows that the LP relaxation of S_v is $\text{conv}(S_v)$. In conclusion, for each $i \in \{0, \dots, n\}$, (IPPR1*) contains all the constraints that describe $\text{conv}(S_i)$. It follows that (IPPR1*) is locally ideal by (3.9). □

We remark that (IPPR1) is a new locally ideal formulation for (3.2). Various other MILP formulations can be obtained by viewing the set of expressions, $E_{I,v} = \{\sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_I:\bar{\boldsymbol{z}}_{(v)}=d_{v,k}} \lambda^{\boldsymbol{z}_I}_{\bar{\boldsymbol{z}}}\}_{k\in[D_v]}$, as special ordered sets of type 2 (SOS2) for all $I \in \mathcal{I}$ and for all $v \in I$. For example, another locally ideal MILP formulation is obtained by removing $\boldsymbol{\rho}$ together with all the constraints containing $\boldsymbol{\rho}$, *i.e.*, (3.3e), (3.3f), and (3.3h), from (IPPR1) and by adding (3.10):

$$\sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I : \bar{\boldsymbol{z}}_{(v)} \leq d_{v,k_2}} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I} \leq \sum_{k \in [k_2]} x_{v,k}, \qquad \forall I \in \mathcal{I}, \forall v \in I, \forall k_2 \in [D_v - 2], \tag{3.10a}$$

$$\sum_{k \in [k_2]} x_{v,k} \leq \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I : \bar{\boldsymbol{z}}_{(v)} \leq d_{v,k_2+1}} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}, \qquad \forall I \in \mathcal{I}, \forall v \in I, \forall k_2 \in [D_v - 2], \tag{3.10b}$$

where (3.10) is inspired from a locally ideal formulation for SOS2 (Kis and Horváth, 2021). We can prove that the obtained formulation is locally ideal using the same decomposition technique used in the proof of Theorem 7. However, introducing $\boldsymbol{\rho}$ in (IPPR1) has the advantage of creating a formulation that is sparser than the formulation with (3.10). Specifically, a single $\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}$ variable appears n times in (3.3e), whereas it may appear at most $n(K-1)$ times in (3.10) where $K := \max_{v \in [n]} D_v$. Given that the number of variables $\boldsymbol{\lambda}$ is large (at most $\sum_{I \in \mathcal{I}} K^{|I|}$), the sparsity of (IPPR1) could prove to be useful for numerical computations. However, we will not pursue this avenue in our computational work, which focuses on exploring the improved tightness obtained from linking constraints discussed in Section 3.3.2. To clearly evaluate the effect of the linking constraints, we will instead continue to use the sharp formulation for (3.2) implemented in ALPINE (Sundar et al., 2021b) and will not introduce the proposed $\boldsymbol{\rho}$ variables.

3.3.2 Linking Constraints

Inspired from (MLP), we next introduce linear equalities in Theorem 9, which we call *linking constraints*, that can be added to (IPPR1) to make it a tighter relaxation of (3.1). To streamline the presentation, we use $\lambda_{d_{v,k}}^{z_v} = \rho_{v,k-1,\text{ub}} + \rho_{v,k,\text{lb}}$ for all $v \in [n]$ and for all $k \in [D_v]$, use $\boldsymbol{\lambda}^{z_v} = \{\lambda_{d_{v,k}}^{z_v}\}_{k \in [D_v]}$ for all $v \in [n]$, and use $\boldsymbol{\lambda}^{\boldsymbol{z}_I} = \{\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}\}_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I}$ for all $I \in \mathcal{I}$.

We next motivate these equalities as providing relationships between degree-$|S|$ multilinear terms of $\{\boldsymbol{\lambda}^{z_v}\}_{v \in S}$ and degree-$|T|$ multilinear terms of $\{\boldsymbol{\lambda}^{z_v}\}_{v \in T}$, for given $S \subsetneq T \subseteq [n]$. Let $S = \{v_1, \ldots, v_s\}$ and $T = \{v_1, \ldots, v_t\}$ with $t > s$. For $k_1 \in [D_{v_1}]$, …,

$k_s \in [D_{v_s}]$, it holds that

$$\prod_{i=1}^{s} \lambda^{z_{v_i}}_{d_{v_i,k_i}} = \left(\prod_{i=1}^{s} \lambda^{z_{v_i}}_{d_{v_i,k_i}} \right) \prod_{i=s+1}^{t} \left(\sum_{k \in [D_{v_i}]} \lambda^{z_{v_i}}_{d_{v_i,k}} \right) = \sum_{k_{s+1}=1}^{D_{s+1}} \cdots \sum_{k_t=1}^{D_t} \prod_{i=1}^{t} \lambda^{z_{v_i}}_{d_{v_i,k_i}}, \tag{3.11}$$

where the first equality holds by definition and (3.4f), and the second is obtained by expanding the expression. Using (3.5) and (3.11), we obtain the following constraints that link $\boldsymbol{\lambda}^{\boldsymbol{z}_{T_1}}$ and $\boldsymbol{\lambda}^{\boldsymbol{z}_{T_2}}$:

$$\sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_1} : \hat{\boldsymbol{z}} = \mathrm{proj}_{\boldsymbol{z}_S} \bar{\boldsymbol{z}}} \lambda^{\boldsymbol{z}_{T_1}}_{\bar{z}} = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_2} : \hat{\boldsymbol{z}} = \mathrm{proj}_{\boldsymbol{z}_S} \bar{\boldsymbol{z}}} \lambda^{\boldsymbol{z}_{T_2}}_{\bar{z}}, \quad \forall S \subseteq (T_1 \cap T_2) : |S| > 0, \ \forall \hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S \tag{3.12}$$

for $T_1, T_2 \in \mathcal{I}$. The condition $|S| > 0$ implies $T_1 \cap T_2 \neq \emptyset$ in these equalities. Both sides of (3.12) correspond to degree-$|S|$ multilinear term $\prod_{v \in S} \lambda_{v,k_v}$ where $k_v \in [D_v]$ is such that $d_{v,k_v} = \hat{\boldsymbol{z}}_{(v)}$ for $v \in S$.

Linking relation (3.12) also can be naturally observed through a reformulation-linearization technique presented in (He and Tawarmalani, 2022). The idea is to remove $\rho_{v,D_v,\mathrm{lb}}$ from (MLP) by plugging $\rho_{v,D_v,\mathrm{lb}} = 1 - (\mathbf{1}^\intercal \boldsymbol{\rho}_v - \rho_{v,D_v,\mathrm{lb}})$ for all $v \in [n]$ so that the remaining $\boldsymbol{\rho}$ variables are all independent. Then, for some $I \subseteq [n]$, $\prod_{v \in I} x_v$ is written as

$$\prod_{v \in I} x_v = \sum_{\{v_1,\ldots,v_p\} \subseteq I} \sum_{k_1 \in [D_{v_1}-1]} \cdots \sum_{k_p \in [D_{v_p}-1]} \prod_{i \in [p]} (d_{v_i,k_i} - d_{v_i,D_{v_i}}) \lambda^{z_{v_i}}_{d_{v_i,k_i}}, \tag{3.13}$$

when plugging (3.4c). Expression (3.13) contains multilinear terms of $\{\lambda^{z_v}_{d_{v,k}}\}_{v \in I, k \in [D_v - 1]}$ of degree from 1 to $|I|$. Considering $j_1, j_2 \in [m]$ such that $I_{j_1} \neq I_{j_2}$ and $I_{j_1} \cap I_{j_2} \neq \emptyset$, (3.4d) for $j \in \{j_1, j_2\}$ have common multilinear terms with degree up to $|I_{j_1} \cap I_{j_2}|$, which are $\prod_{i \in [p]} \lambda^{z_{v_i}}_{d_{v_i,k_i}}$ for all non-empty set $\{v_1, \ldots, v_p\} \subseteq I_{j_1} \cap I_{j_2}$ and for all $(k_1, \ldots, k_p) \in \prod_{i \in [p]} [D_{v_i} - 1]$. It follows that the expressions of w_{j_1} and w_{j_2} after linearization of the formulation with independent $\boldsymbol{\rho}$ have common λ variables, *i.e.*, it instinctively implies (3.12).

Theorem 8. *Consider formulation (IPPR1), i.e., all the variables and constraints in* (3.3)*, together with additional variable* $\{\lambda^{\boldsymbol{z}_S}_{\hat{\boldsymbol{z}}}\}_{S \in \mathcal{S}, \hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S}$*, where* $\mathcal{S} = \{S \subseteq [n] \backslash \{\emptyset\} \mid \exists T_1 \neq$

$T_2 \in \mathcal{I} : S \subsetneq T_1 \cap T_2\}$ and the following constraints

$$\lambda^{\boldsymbol{z}_S}_{\hat{\boldsymbol{z}}} = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_T : \hat{\boldsymbol{z}} = \mathrm{proj}_{\boldsymbol{z}_S} \bar{\boldsymbol{z}}} \lambda^{\boldsymbol{z}_T}_{\bar{\boldsymbol{z}}}, \qquad \forall (S,T) \in \mathcal{S} \times \mathcal{I} : S \subsetneq T,\ \forall \hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S. \tag{3.14}$$

We refer this formulation as (LINK1). Denote by ω_1, ω_2, and ω_3 the optimal objective values of (MLP), (LINK1), and (IPPR1). Then, $\omega_1 \leq \omega_2 \leq \omega_3$.

Proof. Clearly, $\omega_2 \leq \omega_3$. Also, it holds that $\omega_1 \leq \omega_2$ because (3.14) is equivalent to (3.12), which is valid to (MLP) by (3.5) in Proposition 7 and (3.11) as discussed. □

The feasible set of (LINK1) can be strictly contained in that of (IPPR1) as we will illustrate in Example 7. The numbers of additional variables and constraints, however, can become large when there are many discretization points on each axis, as a different constraint is imposed for every element of $\bar{\mathcal{Z}}_S$ and for each $(S,T) \in \mathcal{S} \times \mathcal{I}$ such that $S \subsetneq T$. In Theorem 9, we show that, after adding suitable variables, we can reduce the number of constraints necessary to one for each such (S,T). The constraint in question is obtained by aggregating $|\bar{\mathcal{Z}}_S|$ constraints with multiplier $\prod_{v \in S} \hat{\boldsymbol{z}}_{(v)}$ for all $\hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S$.

Theorem 9. *Consider formulation (IPPR1), i.e., all the variables and constraints in (3.3), together with additional variable μ_S for all $S \in \mathcal{S}$ such that $|S| \geq 2$, and the following constraints*

$$\mu_S = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_T} \left(\prod_{v \in S} \bar{\boldsymbol{z}}_{(v)} \right) \lambda^{\boldsymbol{z}_T}_{\bar{\boldsymbol{z}}}, \qquad \forall (S,T) \in \mathcal{S} \times \mathcal{I} : S \subsetneq T, |S| \geq 2. \tag{3.15}$$

Every feasible solution of this formulation, which we refer to as (LINK2), satisfies (3.12) for all $T_1, T_2 \in \mathcal{I}$ such that $T_1 \cap T_2 \neq \emptyset$.

Proof. We refer to the cardinality of the set S used in the description of (3.12) or (3.15) as the degree of this inequality. We prove the result by induction. We show that every feasible solution of (LINK2) satisfies all constraints (3.12) of degree 1. Then we show that, if all constraints (3.12) of degree up to d are satisfied, then all of the constraints (3.12) of degree up to $d+1$ are satisfied as well.

We first argue that all constraints (3.12) of degree 1 are satisfied by (IPPR1), which is a relaxation of (LINK2). Pick any index $(S, T_1, T_2, \hat{\boldsymbol{z}})$ of (3.12) of degree 1, *i.e.*,

$T_1, T_2 \in \mathcal{I}$, $S \subseteq T_1 \cap T_2$, $\hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S$, and $|S| = 1$. We denote the unique element of S by v. The values of the left-hand-side (lhs) and right-hand-side (rhs) of (3.12) can be interpreted as the convex combination weight at $z_v = \hat{\boldsymbol{z}}_{(v)}$ by (3.3c):

$$z_v = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_i}} \bar{\boldsymbol{z}}_{(v)} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_i}} = \sum_{j \in [D_v]} d_{v,j} \left(\sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_i} : \bar{\boldsymbol{z}}_{(v)} = d_{v,j}} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_i}} \right),$$

for $i \in \{1, 2\}$. Constraints (3.3e)–(3.3i) impose that at most two consecutive weights on axis z_v can be positive. It follows that, given the value of z_v, all weights on axis z_v are uniquely determined since each point in a line segment is a unique convex combination of its ending points. Therefore, for all $j \in [D_v]$, it holds that $\sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_1} : \bar{\boldsymbol{z}}_{(v)} = d_{v,j}} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_1}} = \sum_{\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_{T_2} : \bar{\boldsymbol{z}}_{(v)} = d_{v,j}} \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_2}}$. We conclude that all constraints (3.12) of degree 1 are implied by (IPPR1).

For any positive integer $d \leq \max_{S \in \mathcal{S}} |S| - 1$, we next argue that if (LINK2) implies all constraints (3.12) of degree up to d, then all constraints (3.12) of degree up to $d + 1$ are also implied by (LINK2). Consider any such integer d. Assume that all constraints (3.12) of degree up to d are implied by (LINK2). Pick $(S, T_1, T_2) \subseteq \mathcal{S} \times \mathcal{I} \times \mathcal{I}$ such that $S \subseteq T_1 \cap T_2$ and $|S| = d + 1$. We show that (3.12) with indices $(S, T_1, T_2, \hat{\boldsymbol{z}})$ are implied by (LINK2) for all $\hat{\boldsymbol{z}} \in \bar{\mathcal{Z}}_S$. Consider a feasible solution $(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ of (LINK2). For all $v \in [n]$, we denote by k_v the index such that $x_{v,k_v} = 1$. We define $\hat{Q} = \prod_{v \in S} \delta_{v,k_v}$. Clearly, $\hat{Q}$ is a hyper-rectangle in the space of $\boldsymbol{z}_S$. By (3.3e)–(3.3i), for $I \in \{T_1, T_2\}$ and for $\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_I$, $\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_I}$ is zero if $\operatorname{proj}_{\boldsymbol{z}_S} \bar{\boldsymbol{z}} \notin \operatorname{vert} \hat{Q}$. It follows that many constraints (3.12) are satisfied because they reduce to $0 = 0$. The 2^{d+1} constraints (3.12) that do not simplify in $0 = 0$ are those with indices $(S, T_1, T_2, \hat{\boldsymbol{z}})$ such that $\hat{\boldsymbol{z}} \in \operatorname{vert} \hat{Q}$. Consider an edge of $\hat{Q}$ with endpoints $\hat{\boldsymbol{z}}'$ and $\hat{\boldsymbol{z}}''$. Since $\hat{Q}$ is a hyper-rectangle in $\boldsymbol{z}_S$, there exists a unique $v \in S$ such that $\operatorname{proj}_{\boldsymbol{z}_{S \setminus \{v\}}} \hat{\boldsymbol{z}}' = \operatorname{proj}_{\boldsymbol{z}_{S \setminus \{v\}}} \hat{\boldsymbol{z}}''$. It follows that the sum of the lhs (resp. rhs) of (3.12) with indices $(S, T_1, T_2, \hat{\boldsymbol{z}}')$ and $(S, T_1, T_2, \hat{\boldsymbol{z}}'')$ is equal to the lhs (resp. rhs) of (3.12) with $(S \setminus \{v\}, T_1, T_2, \operatorname{proj}_{\boldsymbol{z}_{S \setminus \{v\}}} \hat{\boldsymbol{z}}')$, respectively. Therefore, for every edge of $\hat{Q}$, the aggregation of the two constraints associated with the endpoints of the edge corresponds to (3.12) of degree d, which is satisfied by the inductive hypothesis. We next show in Lemma 8 that if the 2^d constraints of degree d associated with edges are all satisfied and one of the 2^{d+1} constraints of degree $d + 1$ associated with vertices is satisfied, then other $2^{d+1}-1$ constraints of degree $d+1$ are all satisfied.

Lemma 8. *Consider a hyper-rectangle Q in $\mathbb{R}^d$ for some positive integer d. Consider two vectors $\boldsymbol{x} = \{x_{\boldsymbol{z}}\}_{\boldsymbol{z}\in \operatorname{vert} Q} \in \mathbb{R}^{2^d}$ and $\boldsymbol{y} = \{y_{\boldsymbol{z}}\}_{\boldsymbol{z}\in \operatorname{vert} Q} \in \mathbb{R}^{2^d}$. Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy*

$$x_{\boldsymbol{z}_1} + x_{\boldsymbol{z}_2} = y_{\boldsymbol{z}_1} + y_{\boldsymbol{z}_2}, \qquad \forall \boldsymbol{z}_1, \boldsymbol{z}_2 \in \operatorname{vert} Q : [\boldsymbol{z}_1, \boldsymbol{z}_2] \text{ is an edge of } Q. \tag{3.16}$$

Assume finally that $x_{\boldsymbol{z}'} = y_{\boldsymbol{z}'}$ for some $\boldsymbol{z}' \in \operatorname{vert} Q$. Then $x_{\boldsymbol{z}} = y_{\boldsymbol{z}}$ for all $\boldsymbol{z} \in \operatorname{vert} Q$.

Proof. Consider $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{2^d}$ that satisfies (3.16) and for which $x_{\boldsymbol{z}'} = y_{\boldsymbol{z}'}$ for some $\boldsymbol{z}' \in \operatorname{vert} Q$. Define a simple graph $G = (V, E)$ where $V = \operatorname{vert} Q$ and $E = \{(\boldsymbol{z}_1, \boldsymbol{z}_2) \mid \boldsymbol{z_1}, \boldsymbol{z_2} \in \operatorname{vert} Q, [\boldsymbol{z}_1, \boldsymbol{z}_2] \text{ is an edge of } Q\}$. We denote by $\operatorname{dist}(\boldsymbol{z}_1, \boldsymbol{z}_2)$ the number of edges of a path with fewest edges in G between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The maximum distance between two vertices is bounded above by d. We prove that $x_{\boldsymbol{z}} = y_{\boldsymbol{z}}$ for all $\boldsymbol{z} \in \operatorname{vert} Q$ by induction on $\operatorname{dist}(\boldsymbol{z}, \boldsymbol{z}')$.

First, it holds that $x_{\boldsymbol{z}} = y_{\boldsymbol{z}}$ for all $\boldsymbol{z} \in \operatorname{vert} Q$ such that $\operatorname{dist}(\boldsymbol{z}, \boldsymbol{z}') = 0$ by the lemma's last assumption. Next, we assume that $x_{\boldsymbol{z}} = y_{\boldsymbol{z}}$ for all $\boldsymbol{z} \in \operatorname{vert} Q$ such that $\operatorname{dist}(\boldsymbol{z}, \boldsymbol{z}') = k$ for some integer $k \in \{0, 1, \ldots, d-1\}$. We argue that $\boldsymbol{x}_{\boldsymbol{z}} = \boldsymbol{y}_{\boldsymbol{z}}$ for all $\boldsymbol{z} \in \operatorname{vert} Q$ such that $\operatorname{dist}(\boldsymbol{z}, \boldsymbol{z}') = k+1$. Pick any $\boldsymbol{z}_1 \in \operatorname{vert} Q$ with $\operatorname{dist}(\boldsymbol{z}_1, \boldsymbol{z}') = k+1$. Let P be a path with fewest edges in G from $\boldsymbol{z}_1$ to $\boldsymbol{z}'$. We denote by $\boldsymbol{z}_2$ the vertex that directly succeeds $\boldsymbol{z}_1$ on P. The inductive hypothesis implies that

$$\boldsymbol{x}_{\boldsymbol{z}_2} = \boldsymbol{y}_{\boldsymbol{z}_2} \tag{3.17}$$

because $\operatorname{dist}(\boldsymbol{z}_2, \boldsymbol{z}') = k$. Further, it holds that

$$\boldsymbol{x}_{\boldsymbol{z}_1} + \boldsymbol{x}_{\boldsymbol{z}_2} = \boldsymbol{y}_{\boldsymbol{z}_1} + \boldsymbol{y}_{\boldsymbol{z}_2} \tag{3.18}$$

because $[\boldsymbol{z}_1, \boldsymbol{z}_2]$ is an edge of Q by the definition of G. Therefore, it follows from (3.17) and (3.18) that $\boldsymbol{x}_{\boldsymbol{z}_1} = \boldsymbol{y}_{\boldsymbol{z}_1}$, which proves the inductive step. □

By Lemma 8, it is sufficient to show that the solution satisfies one of 2^{d+1} constraints (3.12) of degree $d+1$. By applying an affine transformation if necessary, we may assume that $\hat{Q} = [0,1]^{d+1}$. Then, after removing zero-valued λ variables and zero-coefficient terms, (3.15) with indices (S, T_1) and (S, T_2) reduce to $\mu_S = \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_1}}$ and $\mu_S = \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_2}}$, respectively, where $\bar{\boldsymbol{z}} = (1, 1, \ldots, 1) \in \mathbb{R}^{d+1}$. It follows that

$$\lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_1}} = \lambda_{\bar{\boldsymbol{z}}}^{\boldsymbol{z}_{T_2}},$$

which is one of the 2^{d+1} constraints (3.12) of degree $d+1$. It follows, by Lemma 8, that the solution satisfies all other 2^{d+1} constraints (3.12) of degree $d+1$. This completes the proof of the inductive step. □

Although the LP relaxation of (LINK2) is not as tight as that of (LINK1), (3.14) in (LINK1) reduces to $0 = 0$ in the process of branch-and-bound whereas (3.15) in (LINK2) does not. We refer to the variables $\{\mu_S\}_{S \in \mathcal{I}'}$ in (3.15) where $\mathcal{I}' = \{S \subseteq [n] : |S| \geq 2, \exists j \in [m] \text{ with } S \subseteq I_j\}$ as *linking variables.* Similarly, we refer to (3.15) as *linking constraints.* Example 7 shows an instance of (MLP) for which (IPPR1) has feasible solutions that do not satisfy linking constraints.

Example 7. *Consider (3.1) with $m = 2$, $n = 4$, $n_A = 0$, $\boldsymbol{\ell} = \boldsymbol{0}$, $\boldsymbol{u} = \boldsymbol{1}$, $f_1(z_1, z_2, z_3) = z_1 z_2 z_3$, $f_2(z_2, z_3, z_4) = z_2 z_3 z_4$, $I_1 = \{1, 2, 3\}$, $I_2 = \{2, 3, 4\}$, and arbitrary cost vectors $\boldsymbol{c_z} \in \mathbb{R}^4$ and $\boldsymbol{c_w} \in \mathbb{R}^2$. We assume that there is no partition, i.e., we model an individual polyhedral relaxation for each f_1 and f_2 over their domains $[0, 1]^3$. Clearly, both $\boldsymbol{\rho}$ and $\boldsymbol{x}$ are not needed in this case, instead we impose that $\boldsymbol{\lambda}^{(z_1,z_2,z_3)}$ and $\boldsymbol{\lambda}^{(z_2,z_3,z_4)}$ correspond to the vertices of $[0, 1]^3$ for each. (IPPR1) for this problem takes the form:*

$$\max \quad \boldsymbol{c}_{\boldsymbol{z}}^{\intercal}\boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal}\boldsymbol{w} \tag{3.19a}$$

$$s.t. \quad z_k = \sum_{(j_1,j_2,j_3)\in\{0,1\}^3} j_k \lambda_{(j_1,j_2,j_3)}^{(z_1,z_2,z_3)}, \qquad \forall k \in \{1, 2, 3\}, \tag{3.19b}$$

$$z_k = \sum_{(j_2,j_3,j_4)\in\{0,1\}^3} j_k \lambda_{(j_2,j_3,j_4)}^{(z_2,z_3,z_4)}, \qquad \forall k \in \{2, 3, 4\}, \tag{3.19c}$$

$$w_1 = \sum_{(j_1,j_2,j_3)\in\{0,1\}^3} (j_1 j_2 j_3) \lambda_{(j_1,j_2,j_3)}^{(z_1,z_2,z_3)} = \lambda_{(1,1,1)}^{(z_1,z_2,z_3)}, \tag{3.19d}$$

$$w_2 = \sum_{(j_2,j_3,j_4)\in\{0,1\}^3} (j_2 j_3 j_4) \lambda_{(j_2,j_3,j_4)}^{(z_2,z_3,z_4)} = \lambda_{(1,1,1)}^{(z_2,z_3,z_4)}, \tag{3.19e}$$

$$\sum_{(j_1,j_2,j_3)\in\{0,1\}^3} \lambda_{(j_1,j_2,j_3)}^{(z_1,z_2,z_3)} = 1, \tag{3.19f}$$

$$\sum_{(j_2,j_3,j_4)\in\{0,1\}^3} \lambda_{(j_2,j_3,j_4)}^{(z_2,z_3,z_4)} = 1, \tag{3.19g}$$

$$\lambda^{(z_1,z_2,z_3)}_{(j_1,j_2,j_3)} \geq 0, \qquad \forall (j_1,j_2,j_3) \in \{0,1\}^3, \tag{3.19h}$$

$$\lambda^{(z_2,z_3,z_4)}_{(j_2,j_3,j_4)} \geq 0, \qquad \forall (j_2,j_3,j_4) \in \{0,1\}^3. \tag{3.19i}$$

Consider the feasible solution $(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda})$ *of (IPPR1) such that* $\boldsymbol{z} = (0.5, 0.5, 0.5, 1)$, $\boldsymbol{w} = (0, 0.5)$, $\lambda^{(z_1,z_2,z_3)}_{(0,0,1)} = \lambda^{(z_1,z_2,z_3)}_{(1,1,0)} = \lambda^{(z_2,z_3,z_4)}_{(0,0,1)} = \lambda^{(z_2,z_3,z_4)}_{(1,1,1)} = 0.5$, *and other unspecified* λ *variables are all zero. Constraint (3.15) for* $S = \{2,3\}$ *becomes* $\mu_{\{2,3\}} = \lambda^{(z_1,z_2,z_3)}_{(0,1,1)} + \lambda^{(z_1,z_2,z_3)}_{(1,1,1)}$ *and becomes* $\mu_{\{2,3\}} = \lambda^{(z_2,z_3,z_4)}_{(1,1,0)} + \lambda^{(z_2,z_3,z_4)}_{(1,1,1)}$, *for* $T = \{1,2,3\}$ *and* $T = \{2,3,4\}$, *respectively. These equalities cannot be satisfied simultaneously by the aforementioned solution, and hence cut it off.*

3.3.3 Computational Results Using Linking Constraints

ALPINE is an open-source MINLP solver that uses PPRs over regular hyper-rectangular partitions. It iteratively solves (3.2) by refining the partition with more discretization points. We implement (3.15) inside of ALPINE. Since the formulation used in ALPINE, see (Sundar et al., 2021a), uses the same $\boldsymbol{\lambda}$, we only additionally add linking variables and constraints.

We consider instances, `mult3` and `mult4` from Los Alamos MINLPLib `https://github.com/lanl-ansi/MINLPLib.jl` (Bao et al., 2015) which are collections of multilinear and polynomial optimization problem instances whose nonlinear terms have degrees up to 3 and 4, respectively. We focus on these instances because the proof of Theorem 9 establishes that linking constraints are not implied in ALPINE's formulation when there is a multilinear function with degree at least 3 in the optimization problem,

We perform the computational experiments on a computer running Linux Mint 19.3 with Intel i7-6700K CPU cores running at 4.00GHz and 48GB of memory. The code is written in Julia v1.6.3 with JuMP package v0.21.10 (Dunning et al., 2017) and ALPINE package v0.2.7. We use IPOPT v3.13.4 (Wächter and Biegler, 2006) and Gurobi v9.0.3 (Gurobi Optimization, LLC, 2021) as the ALPINE's nonlinear and MIP solvers, respectively.

Table 3.1 displays the number of instances for which the first polyhedral relaxation model obtains the optimal objective value as bound. It also presents the average gap of the first relaxation. Table 3.2 displays average of solution times of three global algorithms for solving these instances. The first one is the implementation of linking constraints within ALPINE, the second is the default version of ALPINE, and the third is SCIP (Gleixner et al., 2018). The computation shows that substantial numerical benefits can be achieved for all types of instances from using linking constraints. In particular, linking constraints reduce by a factor 3 the number of instances that cannot be solved within one hour compared to the default version of ALPINE. Figure 3.1 displays the performance profile (Dolan and Moré, 2002) of solution times.

Table 3.1: Tightness of the first relaxation models. Each row displays the result on N instances with same maximum degree (d). N_{solved} is the number of instances for which both algorithms (with/without linking constraints) solve the first relaxation within the time limit. N_{new} is the number of instances that the first relaxation model with linking constraints closes the gap. N_{Alpine} is the number of instances that the first relaxation model without linking constraints closes the gap. GapClosed is the average of gaps closed among instances that are solved and ALPINE does not close the gap.

Type	d	N	N_{solved}	N_{new}	N_{Alpine}	GapClosed (%)
mult	3	60	60	51	5	99.8
mult	4	60	51	44	3	98.0
poly	3	60	51	19	3	94.3
poly	4	40	20	3	0	89.5
Total		220	182	117	11	96.6

Table 3.2: Average solution times of N instances with same maximum degree (d).

Type	d	N	New	Alpine	SCIP
mult	3	60	279.2^{1}	2336.8^{33}	3115.7^{46}
mult	4	60	1205.5^{12}	3370.1^{53}	3564.1^{59}
poly	3	60	1226.0^{12}	2473.1^{32}	3261.6^{53}
poly	4	40	3009.0^{29}	3550.9^{38}	3600.0^{40}
Total		220	1286.4^{54}	2876.5^{156}	3365.8^{198}

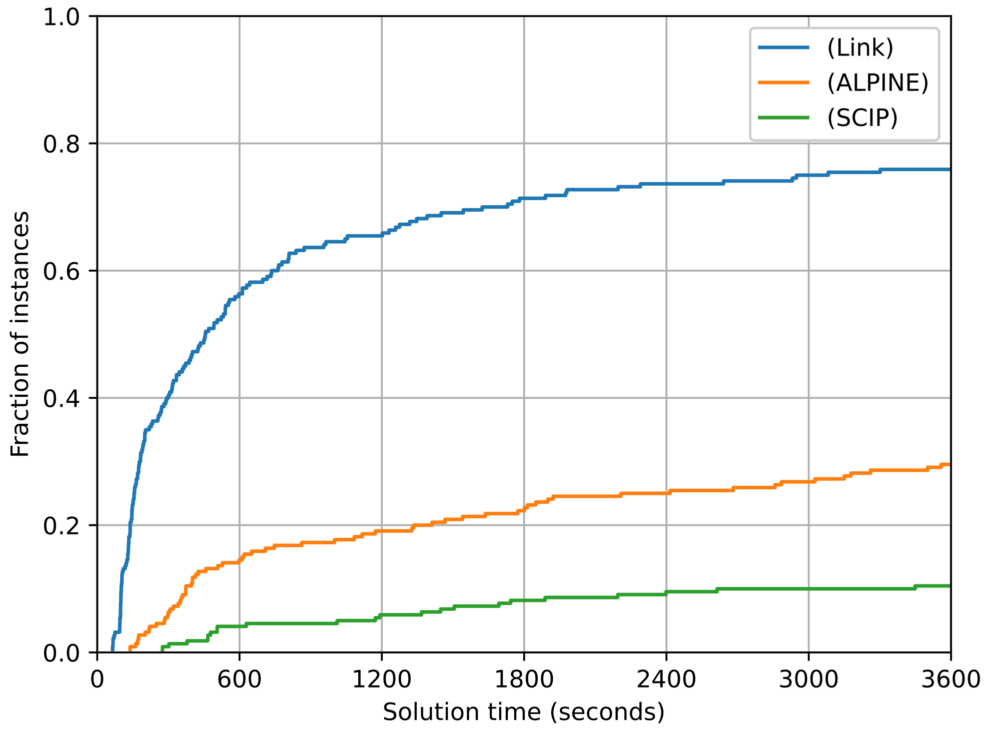


Figure 3.1: Performance profile of solution times.

3.4 Mixed-integer Linear Formulations Over Non-regular Partitions

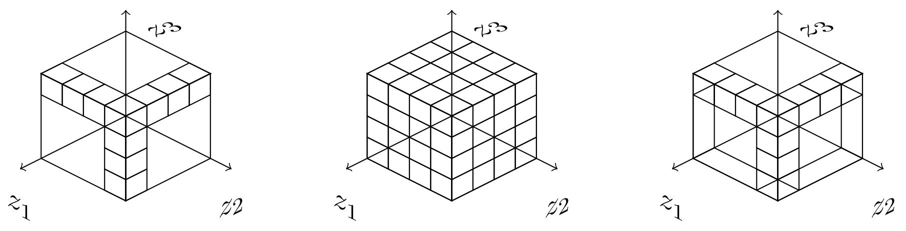
In this section, we introduce MILP formulations for (3.2) for the general case where hyper-rectangular partitions can be non-regular. We first present in Section 3.4.1 a motivating example that shows that non-regular partitions can be substantially more economical than regular partitions in terms of the number of hyper-rectangles they use. In Section 3.4.2, we provide a polynomially-sized locally ideal MILP formulation for any PPR of a single (nonlinear) function over a polyhedral partition of the domain. In Section 3.4.3, we introduce new MILP formulations for (3.2) over (non-regular) hyper-rectangular partitions that combine the advantages of the formulation described in

Section 3.3.1 and the formulation described in Section 3.4.2. Finally, in Section 3.4.4, we perform computational experiments that show that non-regular partitions have distinct computational advantages compared to regular partitions.

3.4.1 Example of an Economical Non-Regular Partition

We say that a set of polytopes, $\{Q'_\ell\}_{\ell\in[L']}$, *refines* another set of polytopes, $\{Q_\ell\}_{\ell\in[L]}$, (or equivalently that $\{Q'_\ell\}_{\ell\in[L']}$ is a *refinement* of $\{Q_\ell\}_{\ell\in[L]}$) if, for all $\ell \in [L]$, there exists $S_\ell \subset [L']$ such that $\bigcup_{k\in S_\ell} Q'_k = Q_\ell$. We introduce next in Example 8 a partition that we use in this section to demonstrate that non-regular refinements are more economical than regular refinements.

Example 8. *Let $n \geq 2$ and $K \geq 2$ be integer parameters. Consider the following partition of $[0,K]^n$. Define unit hyper-cube $Q_{v,j} = \{\boldsymbol{z} \in [0,K]^n \mid z_v \in [j-1,j], z_{v'} \in [K-1,K], \forall v' \in [n]\setminus\{v\}\}$ for all $v \in [n]$ and for all $j \in [K-1]$. Define unit hyper-cube $Q_{\star,K} = [K-1,K]^n$. Define $\mathcal{Q} = \{Q_{v,j}\}_{\forall v\in[n],\forall j\in[K-1]} \cup \{Q_{\star,K}\}$. Finally, define $\hat{Q} := \mathrm{cl}\left([0,K]^n \setminus \bigcup_{Q\in\mathcal{Q}} Q\right)$ to be the closure of the part of $[0,K]^n$ that is not covered by $\mathcal{Q}$. We refer to $\hat{Q}$ as the leftover region of $[0,K]^n$. Collection $\mathcal{Q}' = \mathcal{Q} \cup \{\hat{Q}\}$ is a partition whose cardinality is $n(K-1)+2$. Figure 3.2a graphically depicts $\mathcal{Q}'$ when $n = 3$ and $K = 4$.*



(a) (b) (c)

Figure 3.2: Example partition (a), regular refinement (b), and non-regular refinement (c) when $n = 3$ and $K = 4$.

It is clear that any regular hyper-rectangular partition that refines the partition $\mathcal{Q}'$ of Example 8 requires at least K^n hyper-rectangles. We show in Proposition 8 that there exists a non-regular hyper-rectangular refinement $\{Q'_i\}_{i\in[L']}$ of $\mathcal{Q}'$ with $L' = \mathcal{O}(n^2+nK)$.

Proposition 8. *For any integer parameter $n \geq 2$ and $K \geq 2$, there is a (non-regular) hyper-rectangular partition with cardinality $\frac{n(n-3)}{2} + nK + 1$ that refines the partition $\mathcal{Q}'$ of Example 8.*

Proof. We prove the statement by defining a partition $\mathcal{R}$ of the leftover region $\hat{Q}$ whose cardinality is $\binom{n}{2}$. We define $\mathcal{R} = \{R_{v_1,v_2}\}_{v_1<v_2\in[n]}$ where

$$R_{v_1,v_2} = \{\boldsymbol{z} \in [0,K]^n \mid z_{v_1}, z_{v_2} \leq K-1, z_v \geq K-1, \forall v \in [v_2-1] : v \neq v_1\}, \tag{3.20}$$

for all $v_1, v_2, \in [n]$ such that $v_1 < v_2$.

Clearly, every R_{v_1,v_2} in $\mathcal{R}$ is a hyper-rectangle. We show that $\mathcal{R}$ is a partition of $\hat{Q}$, *i.e.*, (i) $\bigcup_{R\in\mathcal{R}} R = \hat{Q}$ and (ii) the interior of $R_{a,b}$ and $R_{c,d}$ in $\mathcal{R}$ are disjoint if $a \neq c$ or $b \neq d$. We first prove (i). We can write $\hat{Q} = \{\boldsymbol{z} \in [0,K]^n \mid |\{v \in [n] \mid z_v \leq K-1\}| \geq 2\}$. It follows that $\bigcup_{R\in\mathcal{R}} R \subseteq \hat{Q}$. Also, $\hat{Q} \subseteq \bigcup_{R\in\mathcal{R}} R$ because for a point $\boldsymbol{z} \in \hat{Q}$, $\boldsymbol{z} \in R_{v_1,v_2}$ where v_1, v_2 are the first two indices for which $z_v \leq K-1$. We next prove (ii). Consider $R_{a,b}$ and $R_{c,d}$ with $a < b$ and $c < d$. We consider two cases. First, suppose that $a \neq c$. Without loss of generality, assume that $a < c$. Then, hyper-plane $z_a = K-1$ separates $R_{a,b}$ and $R_{c,d}$ as $R_{a,b}$ satisfies $z_a \leq K-1$ and $R_{c,d}$ satisfies $z_a \geq K-1$. Second, suppose that $a = c$. Without loss of generality, assume that $b < d$. Then, hyper-plane $z_b = K-1$ separates $R_{a,b}$ and $R_{c,d}$ as $R_{a,b}$ satisfies $z_b \leq K-1$ and $R_{c,d}$ satisfies $z_b \geq K-1$. Therefore, the interior of any distinct sets $R_{a,b}$ and $R_{c,d}$ in $\mathcal{R}$ are disjoint. In conclusion, $\mathcal{Q} \cup \mathcal{R}$ forms a hyper-rectangular partition of $[0,K]^n$ with cardinality $|\mathcal{Q}| + |\mathcal{R}| = (n(K-1)+1) + \frac{n(n-1)}{2} = \frac{n(n-3)}{2} + nK + 1$. □

Modeling a piecewise function h defined over the partition in Example 8 using a regular partition that refines it, requires at least $(K+1)^n$ variables $\boldsymbol{\lambda}$ (or the number of vertices used in a convex combination). However, Proposition 8 implies that this function can be modeled using a number of λ variables that is at most $2^n(\frac{n(n-3)}{2} + nK + 1)$, if we consider non-regular partitions instead. In practice, n, which is the dimension of the domain of h, can be assumed to be small (≤ 5) as factorable relaxations techniques can be used to express complex functions using simpler atoms. Thus, we may reasonably assume that n is constant. Then, the number of $\boldsymbol{\lambda}$ variables for the regular partition is $\mathcal{O}(K^n)$, whereas that for the non-regular partition is $\mathcal{O}(K)$.

3.4.2 MILP formulations Over the Union of Polytopes

In this section, we provide an MILP formulation for a PPR of a single function $g(\boldsymbol{z})$ over a polyhedral partition of $\mathcal{Z}$. We remark that $g(\boldsymbol{z})$ does not need to be multilinear and the partition does not need to be hyper-rectangular for the results introduced in this section. Let $\{Q_i\}_{i\in[L]}$ be the polyhedral partition of $\mathcal{Z}$ and let $\{\bar{Q}_i\}_{i\in[L]}$ be a PPR of h over $\{Q_i\}_{i\in[L]}$. We next provide an MILP formulation for $(\boldsymbol{z}, w) \in \bigcup_{i\in[L]} \bar{Q}_i$.

We use four types of decision variables: $\boldsymbol{z}$, w, $\boldsymbol{y}$, and $\boldsymbol{\lambda}$. Variables $(\boldsymbol{z}, w) \in \mathbb{R}^{n+1}$ represent points in $\bigcup_{i\in[L]} \bar{Q}_i$. Variable $\lambda_{(\bar{\boldsymbol{z}},\bar{w})}$ represents the convex combination weight for $(\bar{\boldsymbol{z}}, \bar{w})$ for all $(\bar{\boldsymbol{z}}, \bar{w}) \in \bar{\mathcal{Z}} := \bigcup_{i\in[L]} \operatorname{vert} \bar{Q}_i$. We denote by $n_\lambda = |\bar{\mathcal{Z}}|$ the number of λ variables. Variables $\boldsymbol{y} = \{y_i\}_{i\in[L]} \in \Delta^L_{0,1}$ form a unit vector whose single index i taking the value 1 carries the information that $\boldsymbol{z} \in Q_i$. When $\boldsymbol{y} = e_i$, we will refer to Q_i as the *active polytope*.

Variables $\boldsymbol{\lambda}$ and $\boldsymbol{y}$ satisfy the following disjunction:

$$(\boldsymbol{\lambda}, \boldsymbol{y}) \in \bigvee_{i\in[L]} \left\{ (\boldsymbol{\lambda}, \boldsymbol{y}) \in \Delta^{n_{\boldsymbol{\lambda}}} \times \Delta^L_{0,1} \;\middle|\; \begin{array}{l} y_i = 1, \\ \lambda_{(\bar{\boldsymbol{z}},\bar{w})} = 0, \forall (\bar{\boldsymbol{z}}, \bar{w}) \in \bar{\mathcal{Z}} \setminus \operatorname{vert} \bar{Q}_i \end{array} \right\}. \tag{3.21}$$

To derive linear constraints for (3.21), we define bipartite graph $G = (U, V, E)$ where $U = \bar{\mathcal{Z}}$, $V = [L]$, and there exists an edge between $(\bar{\boldsymbol{z}}, \bar{w}) \in U$ and $i \in V$ if $(\bar{\boldsymbol{z}}, \bar{w}) \in \bar{Q}_i$. We denote by $N_G(u) = \{v \in V \mid (u, v) \in E\}$ for $u \in U$ the set of the neighbors of node u in G. Similar to our derivations in Section 2.5.1, a convex hull description of (3.21) can be derived using Hoffman's circulation theorem (Hoffman, 1976). This description is comprised of the following constraints:

$$\sum_{u\in U: N_G(u)\subseteq S} \lambda_u \le \sum_{i\in S} y_v, \qquad \forall S \subsetneq V : S \neq \emptyset. \tag{3.22}$$

We next present an MILP formulation for $(\boldsymbol{z}, w) \in \bigcup_{i\in[L]} \bar{Q}_i$. We use (3.23b)–(3.23f) instead of (3.22) to describe the convex hull of (3.21) using additional variables $\{h_e\}_{e\in E}$ since the number of variables and constraints this formulation requires is polynomial in the number of vertices and edges of G:

$$\begin{pmatrix} \boldsymbol{z} \\ w \end{pmatrix} = \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}} \begin{pmatrix} \bar{\boldsymbol{z}} \\ \bar{w} \end{pmatrix} \lambda_{(\bar{\boldsymbol{z}},\bar{w})}, \tag{3.23a}$$

$$\sum_{e=(u',v')\in E: u'=u} h_e = \lambda_u \qquad \forall u \in U, \tag{3.23b}$$

$$\sum_{e=(u',v')\in E: v'=v} h_e = y_i \qquad \forall v \in V, \tag{3.23c}$$

$$\boldsymbol{\lambda} \in \Delta^{n_\lambda}, \tag{3.23d}$$

$$h_e \geq 0, \qquad \forall e \in E, \tag{3.23e}$$

$$\boldsymbol{y} \in \Delta^L_{0,1}. \tag{3.23f}$$

An MILP formulation for $(\boldsymbol{z}, w) \in \bigcup_{i\in[L]} \bar{Q}_i$ is said to be *ideal* if every extreme point of its LP relaxation complies with the corresponding integrality requirement of the formulation. Formulation (3.23) is ideal because $\boldsymbol{z}$ and w are dependent on $\boldsymbol{\lambda}$ and (3.23b)–(3.23f) describe the convex hull of the system of $\boldsymbol{\lambda}$ and $\boldsymbol{y}$. We say a formulation for $(\boldsymbol{z}, w) \in \bigcup_{i\in[L]} \bar{Q}_i$ is *polynomially-sized* if the total number of variables and constraints is bounded above by a polynomial in L and n_λ. Formulation (3.23) is polynomially-sized.

We believe that formulation (3.23) has advantages over the formulations that can be obtained for this set using results in the literature in that it is polynomially-size and ideal. For instance, consider the formulations for piecewise linear functions over the union of polyhedra presented in (Vielma et al., 2010). Two of these formulations utilize convex combination variables, which are similar to variable $\boldsymbol{\lambda}$ in our formulation.

The first formulation is called the *aggregated convex combination* model. It is a polynomially-sized formulation using variables $(\boldsymbol{x}, w, \boldsymbol{\lambda}, \boldsymbol{y})$ also used in (3.23). It is also a sharp formulation, where a formulation for $(\boldsymbol{z}, w) \in \bigcup_{i\in[L]} \bar{Q}_i$ is said to be *sharp* if the projection of its feasible set over the space of $(\boldsymbol{z}, w)$ variables is $\text{conv}(\bigcup_{i\in[L]} \bar{Q}_i)$. Clearly, every ideal formulation is sharp but the opposite direction does not need to hold. It follows that, when constructing an MIP model using multiple PPRs for different components, using sharp formulations often results in a weaker LP relaxation than using ideal formulations.

The second formulation is called the *disaggregated convex combination* model. It is a polynomially-sized ideal formulation that introduces separate λ variables for the same $\bar{\boldsymbol{z}}$ if multiple $\bar{Q}_i$ share $\bar{\boldsymbol{z}}$ as their vertex. In the context of this chapter, especially when $\{Q_i\}_{i\in[L]}$ is a hyper-rectangular partition of $S \subseteq \mathbb{R}^n$, it introduces up to 2^n variables for the same vertex, which can be much larger than the number of variables we use.

Hence, formulation (3.23) has significant potential advantages for the solution of (3.2) that we develop further in the next section.

3.4.3 MILP Formulations Over Non-Regular Partitions

In this section, we introduce a novel MILP formulation for (3.2) that alleviates some of the disadvantages of formulations in Section 3.3.1 and Section 3.4.2, which can also be used to formulate this problem.

Consider first (IPPR1) in Section 3.3.1. This formulation assumes that partitions are regular. It uses positioning variables $\boldsymbol{x}$ that have the advantage of modeling the geometry of the problem. In the literature, variables $t_{v,j} = \sum_{j'\leq j} x_{v,j'}$, referred to as *incremental variables*, are often used instead of $\boldsymbol{x}$. Incremental variables tend to lead to better branching decisions, guided by the geometry of the problem, because $t_{v,j}$ takes value 1 (resp. 0) only if $z_v \leq d_{v,j+1}$ (resp. $z_v \geq d_{v,j+1}$). Although we can always obtain a regular hyper-rectangular partition by refining a given nonregular hyper-rectangular partition, we have demonstrated in Example 8 that such construction might require K^n hyper-rectangles while the given non-regular partition might only be composed of $\frac{n(n-3)}{2} + nK + 1$ hyper-rectangles.

Consider second formulation (3.23) in Section 3.4.2. In situations such as those illustrated in Example 8, this formulation can be used to avoid generating exponentially many hyper-rectangles. However, it is not clear how to connect different partitions $\{Q_{j,i}\}_{i\in[L_i]}$ as it does not contain the positioning variables $\boldsymbol{x}$ introduced for regular partitions in Section 3.3.1.

We therefore introduce an MILP formulation next for (3.2) that combines the advantages of the formulations described in previous sections by using both $\boldsymbol{x}$ and $\boldsymbol{y}$ variables. This MILP formulation, which we refer to as (IPPR2), applies when $\{Q_{j,i}\}_{i\in[L_j]}$ is a hyper-rectangular partition of $\mathcal{Z}_{I_j}$ for all $j \in [m]$. We use six types of decision variables, $\boldsymbol{z}$, $\boldsymbol{w}$, $\boldsymbol{\lambda}$, $\boldsymbol{h}$, $\boldsymbol{y}$, and $\boldsymbol{x}$. Variables $\boldsymbol{z}$ and $\boldsymbol{w}$ are the same variables used in (3.1). For $j \in [m]$, we denote by $\bar{\mathcal{Z}}_j = \bigcup_{i\in[L_j]} \operatorname{vert} Q_{j,i}$ the set of all vertices used in a convex combination that expresses w_j. Variable $\lambda^j_{\bar{\boldsymbol{z}}}$ indicates the convex combination weight for vertex $\bar{\boldsymbol{z}}$ in the space of $\boldsymbol{z}_{I_j}$ for all $j \in [m]$ and for all $\bar{\boldsymbol{z}} \in \bar{\mathcal{Z}}_j$. Binary variable $y_{j,i}$ is 1 if and only if $Q_{j,i}$ is active among $\{Q_{j,i}\}_{i\in[L_j]}$. Similar to Section 3.4.2, for each $j \in [m]$, we construct bipartite graph $G_j = (U_j, V_j, E_j)$ where $U_j = \bar{\mathcal{Z}}_j$, $V_j = [L_j]$, and $E_j = \{(\bar{\boldsymbol{z}}, i) \in U_j \times V_j \mid \bar{\boldsymbol{z}} \in \operatorname{vert} Q_{j,i}\}$. Then, variable $\boldsymbol{h} = \{h_{j,e}\}_{\forall j\in[m],\forall e\in E_j}$ is used to relate $\boldsymbol{\lambda}$ and $\boldsymbol{y}$. Finally, binary variable $x_{v,k}$ indicates the k^{th} interval on the z_v-axis for all $v \in [n]$ and for all $k \in [D_v]$, where the discretization points $\{d_{v,k}\}_{k\in[D_v]}$ are collected from all partitions $\{Q_{j,i}\}_{i\in[L_j]}$ for all $j \in [m]$.

We relate $\boldsymbol{x}$ and $\boldsymbol{y}_j = \{y_{j,i}\}_{i\in[L_j]}$ which independently indicate the active hyper-rectangle among $\{Q_{j,i}\}_{i\in[L_j]}$ for a fixed $j \in [m]$. For a hyper-rectangle $Q_{j,i}$, we define $k_1(j,i,v) = \min\{k \in [D_v - 1] \mid \delta_{v,k} \in \operatorname{proj}_{z_v} Q_{j,i}\}$ and $k_2(j,i,v) = \max\{k \in [D_v - 1] \mid \delta_{v,k} \in \operatorname{proj}_{z_v} Q_{j,i}\}$ to indicate the most left and right intervals on the z_v-axis that $Q_{j,i}$ overlaps, respectively. Variables $\boldsymbol{x}$ and $\boldsymbol{y}_j$ satisfy the following multilinear constraint:

$$(\boldsymbol{x}, \boldsymbol{y}_j) \in \left\{ (\boldsymbol{x}, \boldsymbol{y}_j) \in \prod_{v\in[n]} \Delta_{0,1}^{D_v-1} \times \Delta^{L_j} \;\middle|\; y_{j,i} = \prod_{v\in[n]} \sum_{k=k_1(j,i,v)}^{k_2(j,i,v)} x_{v,k}, \forall i \in [L_j] \right\}. \quad (3.24)$$

Such relationship between $\boldsymbol{x}$ and $\boldsymbol{y}_j$ is a *facial decomposition of the Cartesian product of simplices*. An explicit convex hull description of (3.24) is provided in Theorem 3 in Chapter 2; this description is in fact the system of (3.25g), (3.25j), and (3.25k) with fixed j. We thus obtain the following formulation (IPPR2):

$$\begin{aligned}
\max \quad & \boldsymbol{c}_{\boldsymbol{z}}^\intercal \boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^\intercal \boldsymbol{w} && && (3.25a)\\
\text{s.t.} \quad & A_{\boldsymbol{z}} \boldsymbol{z} + A_{\boldsymbol{w}} \boldsymbol{w} \le \boldsymbol{b}, && && (3.25b)\\
& \boldsymbol{z}_{I_j} = \sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_j} \bar{\boldsymbol{z}} \lambda^j_{\bar{z}}, && \forall j \in [m], && (3.25c)\\
& w_j = \sum_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_j} f_j(\bar{\boldsymbol{z}}) \lambda^j_{\bar{z}}, && \forall j \in [m], && (3.25d)\\
& \sum_{e=(u',v')\in E_j : u'=u} h_{j,e} = \lambda^j_u, && \forall j \in [m], \forall u \in U_j, && (3.25e)\\
& \sum_{e=(u',v')\in E_j : v'=v} h_{I,e} = y_{j,i}, && \forall j \in [m], \forall v \in V_j, && (3.25f)
\end{aligned}$$

$$\sum_{i\in[L_j]:k_1\le k_1(j,i,v),k_2(j,i,v)\le k_2} y_{j,i} \le \sum_{k=k_1}^{k_2} x_{v,k}, \qquad \forall j\in[m],\forall v\in[n], \quad \forall k_1\le k_2\in[D_v-1], \tag{3.25g}$$

$$\boldsymbol{\lambda^j} = \{\lambda^j_{\bar{\boldsymbol{z}}}\}_{\bar{\boldsymbol{z}}\in\bar{\mathcal{Z}}_j} \in \Delta^{|\bar{\mathcal{Z}}_j|}, \qquad \forall j\in[m], \tag{3.25h}$$

$$h_{j,e}\ge 0, \qquad \forall j\in[m],\forall e\in E_j, \tag{3.25i}$$

$$\boldsymbol{y_j}\in\Delta^{L_j}, \qquad \forall j\in[m], \tag{3.25j}$$

$$\boldsymbol{x_v}\in\Delta^{D_v-1}_{0,1}, \qquad \forall v\in[n]. \tag{3.25k}$$

In (3.25), $\boldsymbol{y}_j$ can be relaxed to be continuous because, given the value of $\boldsymbol{x}$, all but one $y_{j,i}$ can be positive. A formulation for (3.2) is said to be *locally ideal* if it is ideal when $m=1$. We say that a formulation is *polynomially-sized* if the numbers of variables and constraints is polynomial in n (the number of independent variables), n_A (the number of linear constraints), and $n_\lambda=\sum_{j\in[m]}|\bar{\mathcal{Z}}_j|$ (the total number of vertices used in convex combinations).

Theorem 10. *(IPPR2) is a polynomially-sized locally ideal formulation for (3.2).*

Proof. We first show that (IPPR2) is polynomially-sized. The numbers of variables $\boldsymbol{z}$ and $\boldsymbol{\lambda}$ are polynomially-sized by definition. The numbers of variables $\boldsymbol{w}$ and $\boldsymbol{y}$ are polynomially-sized because $m\le n_\lambda$. The number of $\boldsymbol{h}$ variables is polynomially-sized because $\sum_{j\in[m]}|E_j|\le\sum_{j\in[m]}L_j|\bar{\mathcal{Z}}_j|=\mathcal{O}(mn_\lambda^2)=\mathcal{O}(n_\lambda^3)$. The number of $\boldsymbol{x}$ variables is polynomially-sized because $\sum_{v\in[n]}(D_v-1)=\mathcal{O}(nn_\lambda)$. Therefore, the total number of variables is polynomially-sized. The total number of constraints is also polynomially-sized because $m, D_v, |U_j|, |V_j|\le n_\lambda$ and $|E_j|\le n_\lambda^2$.

We next show that (IPPR2) is locally ideal. Suppose $m=1$. Consider the set S_1 in the space of $(\boldsymbol{z},\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{h},\boldsymbol{y})$ obtained by retaining all of the constraints containing these variables. Let S_2 be the set in the space of $(\boldsymbol{y},\boldsymbol{x})$ obtained by retaining all of the constraints containing these variables. Observe that every variable/constraint belongs to at least one of S_1 and S_2. Therefore, the feasible set S of (3.25) can be written as

$$S=\{(\boldsymbol{z},\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{h},\boldsymbol{y},\boldsymbol{x})\mid(\boldsymbol{z},\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{h},\boldsymbol{y})\in S_1,(\boldsymbol{y},\boldsymbol{x})\in S_2\}.$$

Set S_1 is integral because (3.23) is ideal. Set S_2 is integral by Theorem 3 in Chapter 2.

Then, S is also integral; see Lemma 6 in Chapter 2, which states that S is integral if both S_1 and S_2 are integral and the common variable $\boldsymbol{y}$ forms a simplex. Therefore, (IPPR2) is ideal when $m = 1$. □

A distinct advantage of our approach is that it allows the incorporation of geometrical information into models defined over non-regular partitions of their domain, without requiring that the partition be first subdivided into one that is regular. The formulations so-produced therefore have the advantage of typically requiring fewer variables without compromising on their convex hull properties. Intuitively, they combine the advantages of previously proposed approaches. In particular, the presence of positioning variables $\boldsymbol{x}$ might prove helpful in guiding branching decisions. With this respect, one could straightforwardly make use of incremental t variables in our formulations, without compromising on their strength, as variables $x_{v,j}$ are related to these variables via the simple linear and invertible transformation, $x_{v,j} = t_{v,j} - t_{v,j-1}$, where $t_{v,0} = 0$. Formulation (IPPR2) is, to the best of the authors' knowledge, the first polynomially-sized locally ideal formulation modeling (3.2) over general hyper-rectangular partitions.

3.4.4 Computational Experiments

In this section, we perform experiments to demonstrate that non-regular partitions have computational advantages compared to regular partitions. We consider *tree ensemble optimization (TEO)* problems (Mišić, 2020). TEO seeks to find values for the input variables of a given tree ensemble model (used for regression) so as to minimize/maximize the prediction value. TEO has been used to find the best combination of compounds to design new drugs (Mišić, 2020) and to find optimal assortments in marketing that maximize profit (Chen and Mišić, 2022).

A tree ensemble model is a collection of decision trees where each decision tree is a piecewise constant function over a (usually non-regular) hyper-rectangular partition. In this experiment, we consider linear regression trees instead of classical decision trees as the elements of the tree ensemble model. A *linear regression tree* (Quinlan et al., 1992; Dobra and Gehrke, 2002) consists of the association of a decision tree together with a linear model for each leaf. Given an input value, its prediction is computed using the linear model associated with the leaf that the point belongs to.

Given a tree ensemble model composed of linear regression trees, we next describe the form of the TEO problem we consider in our numerical experiment. We denote by n the number of input variables of the given tree ensemble model and suppose that the domain of the input variable $\boldsymbol{z}$ is $\mathcal{Z}$. We denote by m the number of trees in the ensemble and by L_j the number of leaves in the j^{th} tree for $j \in [m]$. The set of all leaves of a decision tree corresponds to a hyper-rectangular partition of its domain because each nonleaf node of the tree divides the domain using a hyperplane $z_v = a$ for some $a \in \mathbb{R}$. We denote by $Q_{j,i}$ and $f_{j,i}(\boldsymbol{z})$ the hyper-rectangle and the linear function corresponding to the i^{th} leaf in the j^{th} tree, respectively, for $j \in [m]$ and for $i \in [L_j]$. Using this notation, we formulate the problem as

$$\max \quad \frac{1}{m} \sum_{j \in [m]} w_j \tag{3.26a}$$

$$\text{s.t.} \quad \begin{pmatrix} \boldsymbol{z} \\ w_j \end{pmatrix} \in \bigcup_{i \in [L_j]} \left\{ \begin{pmatrix} \boldsymbol{z} \\ f_{j,i}(\boldsymbol{z}) \end{pmatrix} \;\middle|\; \boldsymbol{z} \in Q_{j,i} \right\}, \qquad \forall j \in [m], \tag{3.26b}$$

$$\boldsymbol{z} \in \mathcal{Z}. \tag{3.26c}$$

When the value of $\boldsymbol{z}$ lays on the boundary of multiple hyper-rectangles, model (3.26) is free to select w_j using any of the corresponding linear functions. This assumption is common in TEO and general piecewise problems. Constraint (3.26b) can be written as $(\boldsymbol{z}, w_j) \in \bigcup_{i \in [L_j]} \operatorname{conv}\{(\boldsymbol{z}, f_{j,i}(\boldsymbol{z}))\}_{\boldsymbol{z} \in \operatorname{vert} Q_{j,i}}$ for all $j \in [m]$ because the functions $f_{j,i}(\boldsymbol{z})$ are linear over $Q_{j,i}$. Then, (3.26) takes the form of a PPR of an optimization problem over a hyper-rectangular partition.

The partition used in the above problem is typically not regular. We could therefore build an alternative formulation by constructing a regular partition $\{Q'_{j,i}\}_{i \in [L'_j]}$ that refines $\{Q_{j,i}\}_{i \in L_j}$ for all $j \in [m]$ where L'_j is a positive integer. Specifically, for $j \in [m]$, $\{Q'_{j,i}\}_{i \in [L'_j]}$ is constructed using the discretization points that appear in the j^{th} decision tree. The linear function $f'_{j,i'}(\boldsymbol{z})$ associated with $Q'_{j,i'}$ is defined as $f_{j,i}$ when $Q'_{j,i'} \subseteq Q_{j,i}$. It is clear that the TEO problem constructed using $\{Q'_{j,i}\}_{i \in [L'_j]}$ and $\{f'_{j,i'}(\boldsymbol{z})\}_{i \in [L'_j]}$ for each $j \in [m]$ is equivalent to (3.26).

We develop a formulation (3.28) that can be applied to the case when partitions are regular and multiple λ variables are used for the same vertex $\bar{\boldsymbol{z}}$. Similar to (IPPR1), it

Table 3.3: Solution times and numbers of hyper-rectangles in regular/non-regular partitions for TEO instances with n input variables and T trees with maximum depth D.

Data set	n	D	T	# of hyper-rectangles		Solution times	
				Non-regular	Regular	Non-regular	Regular
diabetes	10	2	5	20	38	1.7	0.8
diabetes	10	2	10	40	76	9.9	12.3
diabetes	10	2	15	59	112	12.4	64.6
diabetes	10	2	20	77	140	21.3	120.8
diabetes	10	3	5	29	120	2.9	44.4
diabetes	10	3	10	59	224	15.4	23.3
diabetes	10	3	15	88	338	28.1	212.0
diabetes	10	3	20	119	498	35.5	301.7
diabetes	10	4	5	35	322	10.4	384.7
diabetes	10	4	10	72	666	31.2	963.7
diabetes	10	4	15	106	894	63.0	1962.9
diabetes	10	4	20	144	1176	106.2	3600.0
house price	8	2	5	20	40	0.2	0.8
house price	8	2	10	40	80	0.6	1.0
house price	8	2	15	60	120	2.4	6.8
house price	8	2	20	80	158	4.6	13.6
house price	8	3	5	27	72	0.4	3.5
house price	8	3	10	56	176	2.2	21.5
house price	8	3	15	86	296	1.2	11.0
house price	8	3	20	114	380	10.3	65.1
house price	8	4	10	69	432	1.8	55.0
house price	8	4	15	105	664	5.4	139.8
house price	8	4	20	143	890	10.6	225.3

uses $(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{x})$ variables. The full description of (3.28) is available in Section 3.6.

The experiment we carry is as follows. First, we train tree ensemble models with linear regression trees for the diabetes data set from UCI machine learning repository (Dua and Graff, 2017) and for the California house price data set (Pace and Barry, 1997). We use the training algorithm available at `https://github.com/cerlymarco/linear-tree`. We use different maximum depths ($D = 2, 3, 4$) and different number of trees ($T = 5, 10, 15, 20$). Finally, we solve the instances of TEO with formulation (IPPR2) applied to their natural non-regular partitions and with formulation (3.28) applied to the refined regular partition described above.

Table 3.3 summarizes the result of this experiment for both data sets. Each row in the table indicates what data set is being considered, together with the number n of its input variables, the maximum depth D of trees, and the number of trees T in the ensemble. It then displays the numbers of hyper-rectangles used in each of the formulations and their solution times, respectively. We observe that the refined partitions require the introduction of a significant number of hyper-rectangles, number that increases as n or D increases. This observation parallels the discussion of Example 8. In particular, the reason that the number of hyper-rectangles required in regular partitions increases quickly when n or D becomes large is precisely the reason that it increases quickly when the parameters n or K of Example 8 become large. In fact, the number of input variables N naturally maps to the number of dimensions n of Example 8 while larger depths D of trees in the ensemble influence the number of discretization points required, which are captured by the parameter K of Example 8. Table 3.3 also shows that this increase in number of partition elements comes at the cost of significantly longer solution times, especially as parameters n or D become larger.

3.5 Conclusion

In this chapter, we consider PPRs of multilinear optimization problems over (axis-parallel) hyper-rectangular partitions. We provide a new formulation for PPRs over regular partitions using linking constraints. These constraints improve the formulations based on individual polyhedral relaxations that can be found in the literature. We show that an implementation of our relaxation inside of the open-source MINLP solver ALPINE significantly improves its performance on a variety of multilinear and polynomial optimization problem instances from Los Alamos MINLPLib. We also provide the first MILP formulation for PPRs over non-regular partitions. Finally, we perform computational experiments that show that a non-regular partition-based approach can outperform approaches based on refined regular partitions.

3.6 Supplements: A Formulation Used for Experiments in Section 3.4.4

In this section, we provide a formulation for

$$
\begin{aligned}
\max \quad & \boldsymbol{c}_{\boldsymbol{z}}^{\intercal}\boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal}\boldsymbol{w} && && (3.27a)\\
\text{s.t.} \quad & A_{\boldsymbol{z}}\boldsymbol{z} + A_{\boldsymbol{w}}\boldsymbol{w} \le \boldsymbol{b}, && && (3.27b)\\
& \begin{pmatrix} \boldsymbol{z}_{I_j} \\ w_j \end{pmatrix} \in \bigcup_{i\in[L_j]} \bar{Q}_{j,i}, && \forall j \in [m]. && (3.27c)
\end{aligned}
$$

An MILP formulation for (3.27) is described as follows:

$$
\begin{aligned}
\max \quad & \boldsymbol{c}_{\boldsymbol{z}}^{\intercal}\boldsymbol{z} + \boldsymbol{c}_{\boldsymbol{w}}^{\intercal}\boldsymbol{w} && && (3.28a)\\
\text{s.t.} \quad & A_{\boldsymbol{z}}\boldsymbol{z} + A_{\boldsymbol{w}}\boldsymbol{w} \le \boldsymbol{b}, && && (3.28b)\\
& \boldsymbol{z}_{I_j} = \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j} \bar{\boldsymbol{z}}\lambda^j_{\bar{\boldsymbol{z}},\bar{w}}, && \forall j \in [m], && (3.28c)\\
& w_j = \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j} \bar{w}\lambda^j_{\bar{\boldsymbol{z}},\bar{w}}, && \forall j \in [m], && (3.28d)\\
& \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j:\bar{\boldsymbol{z}}_{(v)}\le d_{v,k_2}} \lambda^j_{\bar{\boldsymbol{z}},\bar{w}} \le \sum_{k=1}^{k_2} x_{v,k}, && \forall j \in [m], \forall v \in I_j, \forall k_2 \in [D_v - 2], && (3.28e)\\
& \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j:\bar{\boldsymbol{z}}_{(v)}\ge d_{v,k_1+1}} \lambda^j_{\bar{\boldsymbol{z}},\bar{w}} \le \sum_{k=k_1}^{D_v-1} x_{v,k}, && \forall j \in [m], \forall v \in I_j, \forall k_1 \in [2..(D_v - 1)], && (3.28f)\\
& \sum_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j:\bar{\boldsymbol{z}}_{(v)}\in[d_{v,k_1+1},d_{v,k_2}]} \lambda^j_{\bar{\boldsymbol{z}},\bar{w}} \le \sum_{k=k_1}^{k_2} x_{v,k}, && \forall j \in [m], \forall v \in I_j, && (3.28g)\\
& && \forall k_1 < k_2 \in [2..(D_v - 2)], &&\\
& \boldsymbol{x}_{\boldsymbol{v}} \in \Delta_{0,1}^{D_v-1}, && \forall v \in [n], && (3.28h)\\
& \boldsymbol{\lambda^j} = \{\lambda^j_{\bar{\boldsymbol{z}},\bar{w}}\}_{(\bar{\boldsymbol{z}},\bar{w})\in\bar{\mathcal{Z}}_j} \in \Delta^{|\bar{\mathcal{Z}}_j|}, && \forall j \in [m], && (3.28i)
\end{aligned}
$$

where $\bar{\mathcal{Z}}_j = \{(\bar{\boldsymbol{z}}, \bar{w})\}_{i\in[L_j],(\bar{\boldsymbol{z}},\bar{w})\in\operatorname{vert}\bar{Q}_{j,i}}$ for all $j \in [m]$.

Chapter 4

Learning Symbolic Expressions: Mixed-Integer Formulations, Cuts, and Heuristics

The work described in this chapter was performed under the guidance of Dr. Sven Leyffer and Dr. Prasanna Balaprakash at Argonne National Laboratory. This work was supported by the Applied Mathematics activity within the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

4.1 Preface

In this chapter we consider the problem of learning a regression function without assuming its functional form. This problem is referred to as *symbolic regression*. An expression tree is typically used to represent a solution function, which is determined by assigning operators and operands to the nodes. The symbolic regression problem can be formulated as a nonconvex mixed-integer nonlinear program (MINLP), where binary variables are used to assign operators and nonlinear expressions are used to propagate data values through nonlinear operators such as square, square root, and exponential.

We extend this formulation by adding new cuts that improve the solution of this challenging MINLP. We also propose a heuristic that iteratively builds an expression tree by solving a restricted MINLP. We perform computational experiments and compare our approach with a mixed-integer program-based method and a neural-network-based method from the literature.

4.2 Introduction

We consider the problem of learning symbolic expressions, which is referred to as *symbolic regression*. Symbolic regression is a form of regression that learns functional expressions from observational data. Unlike traditional regression, symbolic regression does not assume a fixed functional form but instead learns the functional relationship and its constants. Given observational data in terms of independent variables, $x_i \in \mathbb{R}^d$, and dependent variables (function values), $z_i \in \mathbb{R}$, for $i = 1, \ldots, n_{\text{data}}$, symbolic regression aims to find the best functional form that maps the x-values to the z-values by solving the following optimization problem:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n_{\text{data}}} \left(f(x_i) - z_i\right)^2 , \tag{4.1}$$

where $\mathcal{F}$ is the space of functions from which f is chosen. We note that other loss functions involving general norms are also possible and that, in general, problem (4.1) is an infinite-dimensional optimization problems. Various applications of symbolic regression have been presented in different fields including materials science (Wang et al., 2019), fluid systems (Duriez et al., 2017), physics (Schmidt and Lipson, 2009; Udrescu et al., 2020; Udrescu and Tegmark, 2020), and civil engineering (Tarawneh et al., 2019).

Symbolic regression is especially useful when we do not know the precise functional form that relates the independent variables x to the dependent variables z or when we wish to exploit the freedom of optimally choosing the functional form. Unlike other forms of regression such as deep neural network, symbolic regression provides interpretable nonlinear functional forms in terms of the independent variables. Given data $(x_i, z_i) \in \mathbb{R}^{d+1}$, $i = 1, \ldots, n_{\text{data}}$ and a set of mathematical operators, symbolic regression searches for a best-fit mathematical expression as a combination of these operators, independent variables, and constant. Given suitable restrictions on the function space $\mathcal{F}$ (e.g., a finite set of mathematical operators), we can formulate (4.1) as a nonconvex mixed-integer nonlinear program (MINLP). In this chapter, we describe new cutting planes to enhance the MINLP formulation, develop new heuristics to solve the resulting MINLP, and demonstrate the effectiveness of our approach on a broad set of test problems.

The remainder of this chapter is organized as follows. In the rest of this section, we review some background material on expression trees and the literature on symbolic regression. In Section 4.3, we introduce a MINLP formulation that improves the existing formulations by adding new sets of cutting planes. In Section 4.4, we introduce the sequential tree construction heuristic for solving the MINLPs arising in symbolic regression. We demonstrate the effectiveness of our ideas in a detailed numerical comparison in Section 4.5, before concluding with some final remarks in Section 4.6.

4.2.1 Review of Expression Trees

A mathematical expression can be represented by an expression tree. Figure 4.1 shows an expression tree of the pendulum formula (in general, an expression tree is not unique). An expression tree can be constructed by assigning an operand (an independent variable (x_j) or constant (cst)) or an operator ($+$, $-$, $*$, $/$, exp, log, $(\cdot)^2$, $(\cdot)^3$, $\sqrt{}$, etc.) to the nodes on a tree.

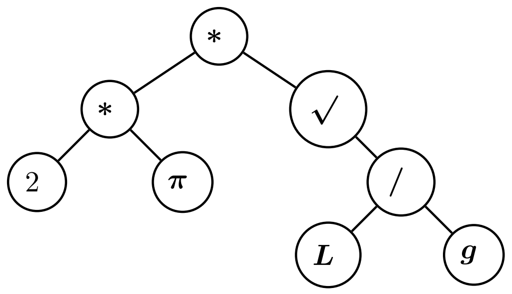


Figure 4.1: An expression tree of the pendulum formula, $T = 2\pi\sqrt{\frac{L}{g}}$.

The objective in (4.1) is to minimize the empirical loss, in other words, to maximize the accuracy of the expression. Additionally, the objective function may include a regularization term modeling the complexity of the expression to obtain a simple expression. A common way to measure the complexity of symbolic regression is to calculate the number of nodes in an expression tree. We review in Section 4.3 how we can formulate (4.1) by modeling expression trees using binary variables.

4.2.2 Methods, Test Problems, and Challenges for Symbolic Regression

Over the past few years, researchers have expressed renewed interest in learning symbolic expressions. Both exact formulations and solution techniques based on mixed-integer nonlinear programming and heuristic search techniques have been proposed. Recently, machine learning methodologies and hybrid techniques have also been proposed. Below, we briefly review each class of methods as well as test problem collections.

Heuristic Techniques for Symbolic Regression Genetic programming is the most common approach for solving symbolic regression. It begins with an initial population of individuals corresponding to randomly selected expression trees. These trees are compared by a fitness measure and error metrics. Individuals with high scores have a higher probability of being selected for the next iteration of crossover, mutation, and reproduction. Ideas to enhance genetic algorithms have been proposed in Kronberger et al. (2019) and Kammerer et al. (2020) to reduce the search space. Nicolau and McDermott (2020) use prior information of the values of the dependent variable. The quality of solutions is not stable, however, because genetic algorithms are a stochastic process, which means that it can generate different solutions for the same input data and the same settings. Kammerer et al. (2020) remark that "it might produce highly dissimilar solutions even for the same input data."

Exact Mixed-Integer Approaches to Symbolic Regression Exact approaches based on the MINLP formulation are deterministic in the sense that they return the same solution if the input data and parameter settings are the same. In principle MINLP approaches are exact in the sense that they will recover the global solution of

(4.1), although their runtime may be prohibitively long in practice. MINLP formulations were first proposed in Cozad (2014), extended in Cozad and Sahinidis (2018) and Neumann et al. (2019), and independently studied in Austel et al. (2017) and Horesh et al. (2016). MINLP formulations use binary variables to define the expression tree and continuous variables to represent intermediate values for each node and each data point; see Section 4.3. The resulting optimization problem is a nonlinear, nonconvex MINLP, because it involves nonlinear operators such as $*, /, \exp$, and $\log$. The solution time typically increases exponentially in the maximum depth of the tree. To limit the runtime of the MINLP solvers, Austel et al. (2017), Cozad (2014), Cozad and Sahinidis (2018), and Neumann et al. (2019) limit the structure of the expression tree (usually by limiting its depth) and solve a smaller MINLP. The approaches in Austel et al. (2017), Cozad (2014), and Cozad and Sahinidis (2018) are tested only on noiseless data. Neumann et al. (2019) propose an interesting methodology to avoid overfitting: (1) they add a constraint to limit the complexity of the expression tree (number of nodes); (2) they then generate a portfolio of solutions by varying the complexity limitation on the training set; and (3) they choose the solution based on the validation error.

Approaches Based on Machine Learning Methodologies An approach based on training a neural network has been proposed in Udrescu and Tegmark (2020) and Udrescu et al. (2020). It is referred to as AI Feynman. A key of AI Feynman is to discover functional properties of the overall function using a neural network in order to reduce the search space and decompose the overall symbolic regression problem into smaller subproblems. The method recursively applies dimension reduction techniques (dimensional analysis, symmetry, and separability detection) until the remaining components are simple enough to be detected by polynomial fits or complete enumeration. These techniques require knowledge of the units of the independent and the dependent variables in order to perform the dimensional analysis, as well as smoothness of the underlying expression, because the method trains a neural network to evaluate values on missing points. A related method is considered in Cranmer et al. (2020), which involves using a graph neural network.

Hybrid Techniques for Symbolic Regression Austel et al. (2020) propose an interesting idea that considers a generalized expression tree instead of an expression tree. A generalized tree assigns a monomial ($hx_1^{a_1}x_2^{a_2}\cdots x_d^{a_d}$) to each leaf node, instead of a single variable or constant. Consequently, generalized expression trees can represent a larger class of functions for the same depth.

The algorithm has two steps. First, it lists all generalized trees up to depth D. The number of generalized trees is reduced by removing redundant expressions. For example, both the generalized tree corresponding to the summation of two monomials ($hx_1^{a_1}x_2^{a_2}\cdots x_d^{a_d}+gx_1^{b_1}x_2^{b_2}\cdots x_d^{b_d}$) and the generalized tree corresponding to the subtraction of two monomials ($h'x_1^{a'_1}x_2^{a'_2}\cdots x_d^{a'_d}-g'x_1^{b'_1}x_2^{b'_2}\cdots x_d^{b'_d}$) are equivalent because both can represent the same set of functions; therefore, one of the expressions can be removed from the list. The list of all generalized trees up to depth one with operators $\{+,-,*,/,\sqrt{}\}$ is

$$L_1,\ \sqrt{L_1},\ (L_1+L_2\equiv L_1-L_2),\ (L_1*L_2\equiv L_1/L_2),$$

where L_1 and L_2 are monomials and $\equiv$ represents that two expressions are equivalent. Second, the algorithm solves a series of optimization problems to find global solutions for each generalized tree. For example, the optimization problem of L_1+L_2 is

$$\min_{h,a_1,\ldots,a_d,g,b_1,\ldots,b_d}\sum_{i=1}^{n_{\text{data}}}\left(z_i-\left(hx_{i,1}^{a_1}x_{i,2}^{a_2}\cdots x_{i,d}^{a_d}+gx_{i,1}^{b_1}x_{i,2}^{b_2}\cdots x_{i,d}^{b_d}\right)\right)^2, \tag{4.2}$$

where h,g are bounded continuous variables and a,b are bounded integer variables. Even though the problem assumes that the tree structure of the generalized expression tree is given, it is a mixed-integer nonlinear (nonconvex) programming problem that is in NP-hard. Constraints are added based on the knowledge of the units of the independent and the dependent variables to reduce the search space.

Benchmark Problems for Symbolic Regression Several test problem collections have been produced to test symbolic regression ideas. For example, ALAMO, the Automatic Learning of Algebraic Models, is a software package for symbolic regression; see `http://minlp.com/alamo`. The test set of Cozad and Sahinidis (2018) (see `http://minlp.com/nlp-and-minlp-test-problems`) has 24 instances. Austel et al. (2017)

consider Kepler's law $d = \sqrt[3]{c\tau^2(M+m)}$ and the period of the pendulum $\tau = 2\pi\sqrt{\ell/g}$. A set of examples from White et al. (2013) is available at http://gpbenchmarks.org. A number of test problems are also in Udrescu and Tegmark (2020) and Udrescu et al. (2020), which are available at https://space.mit.edu/home/tegmark/aifeynman.html.

The Challenges of Symbolic Regression Solving symbolic regression problems such as (4.1) has been shown to be challenging; see, for example, Austel et al. (2020), Austel et al. (2017), Cozad and Sahinidis (2018), and Neumann et al. (2019). In particular, the problem complexity increases with the number of operators and the number of operator types. For example, the number of expression trees represented by d independent variables and B binary operators with a maximum depth D is more than $(B \cdot d)^{2^D}/B$.[1] Another challenge is the non-convexity of the problem, which remains even if the expression tree is fixed because the problem with a fixed expression tree is equivalent to optimize parameters of an arbitrary functional form.

4.3 An Improved MINLP Formulation of Symbolic Regression

We review and improve a MINLP formulation that searches an expression tree with the minimum training error given data points $(x_{i,1}, \ldots, x_{i,d}, z_i) \in \mathbb{R}^{d+1}$ for $i = 1, \ldots, n_{\text{data}}$. Our formulation improves the one proposed by Cozad and Sahinidis (2018). The new constraints remove equivalent expression trees and tighten the feasible set of the relaxation.

4.3.1 Notation

A *relaxation* refers to the relaxation obtained by relaxing binary variables. We let $[a] := \{1, 2, \ldots, a\}$ for $a \in \mathbb{Z}_{++}$, the set of positive integers.

[1]Let T_δ denote the number of expression trees up to depth δ. $T_D \geq (B \cdot d)^{2^D}/B$ is derived from the system of $T_0 \geq n$ and $T_\delta \geq B \cdot T_{\delta-1}^2$. The equality holds if we allow an expression tree to use only binary operators and the independent variables.

4.3.2 Inputs

We are given a set of operators and a set of nodes that define the superset of all feasible expression trees. By limiting the number of nodes and operands used to construct the expression tree, we transform the infinite-dimensional problem (4.1) into a finite-dimensional problem. We denote by $\mathcal{P} \subseteq \{+, -, *, /, \sqrt{}, \exp, \log\}$ a finite set of operators. To streamline our presentation, we define the set of binary operators $\mathcal{B} := \mathcal{P} \cap \{+, -, *, /\}$, the set of unary operators $\mathcal{U} := \mathcal{P} \cap \{\sqrt{}, \exp, \log\}$, and the set of operands $\mathcal{L} = \{x_1, \ldots, x_d, \mathrm{cst}\}$, where cst is a constant. We denote the set of all operators and operands by $\mathcal{O} := \mathcal{B} \cup \mathcal{U} \cup \mathcal{L}$. Although we only consider seven operators in this chapter, formulations for an extended operator set including other unary or binary operators can be easily derived using the techniques described here.

We identify each node of the tree by an integer, and we let $\mathcal{N}$ denote the set of all nodes in the tree. We denote the children of node $n \in \mathcal{N}$ by $2n$ and $2n+1$, respectively. We assume $1 \in \mathcal{N}$, which is the root of the tree. We denote by $\mathcal{T}$ the set of terminal nodes (that have no child). We assume that $\mathcal{N}$ corresponds to a full binary tree.[2] For example, $\mathcal{N} = \{1, 2, 3, 6, 7\}$ is a full binary tree, while $\mathcal{N} = \{1, 2, 3, 4\}$ is not because node 2 has only one child, namely, node 4. Provided that the set of operators $\mathcal{O}$ is finite and we limit the number of nodes, this MINLP formulation transforms the infinite-dimensional functional approximation (4.1) into a finite-dimensional problem.

4.3.3 Decision Variables and Individual Variable Restrictions

There are three types of decision variables. The binary variable y_n^o is one if operator o is assigned to node n, and zero otherwise. Variable c_n is the constant value at node n if node n exists, and zero otherwise. Therefore, y_n^o and c_n determine the expression tree. Variable $v_{i,n}$ represents the intermediate computation value at node n for data point i. In other words, $v_{i,n}$ is the value of the symbolic expression represented by the subtree rooted by node n at data point i. Therefore, $v_{i,1}$ is the value predicted by the expression tree of data point i. All continuous variables are bounded. To streamline our presentation, we use $n \notin \mathcal{T}$ instead of $n \in \mathcal{N} \setminus \mathcal{T}$ in the following discussions. We define $\mathcal{Y} := \{(n, o),\ \forall o \in \mathcal{O},\ \forall n \notin \mathcal{T}\} \cup \{(n, o),\ \forall o \in \mathcal{L},\ \forall n \in \mathcal{T}\}$, the set of all pairs of

[2] A full (proper) binary tree is a tree in which every node has zero or two children.

node n and operator o such that o can be assigned to n. To summarize, our model has the following set of variables and ranges:

$$y_n^o \in \{0,1\}, \qquad \forall (n,o) \in \mathcal{Y}, \tag{4.3a}$$

$$c_{\text{lo}} \le c_n \le c_{\text{up}}, \qquad \forall n \in \mathcal{N}, \tag{4.3b}$$

$$v_{\text{lo}} \le v_{i,n} \le v_{\text{up}}, \qquad \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}. \tag{4.3c}$$

We assume without loss of generality that $v_{\text{lo}} \le c_{\text{lo}} \le 0 \le c_{\text{up}} \le v_{\text{up}}$.

4.3.4 Objective Function

We minimize the mean of the squared errors

$$\min \quad \frac{1}{n_{\text{data}}} \sum_{i=1}^{n_{\text{data}}} (z_i - v_{i,1})^2. \tag{4.4}$$

Additionally, we might add a regularization term such as $\lambda \sum_{(n,o)\in\mathcal{Y}} y_n^o$, where $\lambda \in \mathbb{R}_+$ is a regularization parameter to promote a sparser expression tree.

4.3.5 Constraints

In Section 4.3.5, we introduce constraints that are necessary to solve this problem. In Section 4.3.5, we introduce constraints that remove equivalent expression trees and/or reduce the space of the relaxation, which potentially lead to an improvement in computation. We compare the constraints with the similar constraints in Cozad and Sahinidis (2018) in each section. For the sake of completeness, the formulation proposed in Cozad and Sahinidis (2018) is summarized in Section 4.7 in terms of our notation.

Tree-Defining Constraints

Tree-defining constraints enforce that the assignment of operators and operands results in a valid expression tree. The constraints consist of (4.5), (4.19a), and (4.19b). In addition to (4.19a) and (4.19b) from Cozad and Sahinidis (2018), we use the following tree-defining constraints:

$$\sum_{o \in \mathcal{B} \cup \mathcal{U}} y_n^o = \sum_{o \in \mathcal{O}} y_{2n+1}^o, \qquad n \notin \mathcal{T}, \tag{4.5a}$$

$$\sum_{o \in \mathcal{B}} y_n^o = \sum_{o \in \mathcal{O}} y_{2n}^o, \qquad n \notin \mathcal{T}. \tag{4.5b}$$

Constraint (4.5a) enforces that a binary/unary operator is assigned to node n if and only if its right child (node $2n+1$) exists. Constraint (4.5b) enforces that a binary operator is assigned to node n if and only if its left child (node $2n$) exists. Constraint (4.19a) forces the assignment of at most one operator to a node. Constraint (4.19b) forces the expression tree to include at least one independent variable. All four constraints are necessary to obtain an expression tree with valid operator/operand assignments.

In contrast, the tree-defining constraints from Cozad and Sahinidis (2018) are (4.19a)-(4.19f). Figure 4.2 shows two assignments of operators and operands to an expression tree. Both represent the symbolic expression x_1, and both are feasible in (4.19c)-(4.19f), but only Figure 4.2a is feasible in (4.5). We formalize this observation by showing in

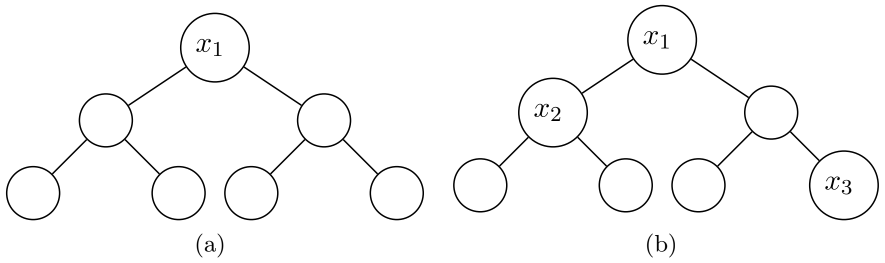


Figure 4.2: Two equivalent expression trees corresponding to x_1. Both are feasible in the Cozad and Sahinidis (2018)'s formulation, while only (a) is feasible in our improved formulation.

Lemma 9 that our tree-defining constraints system is tighter.

Lemma 9. *It holds that* $\{y \in \{0,1\}^{|\mathcal{Y}|} \mid (4.5), (4.19a), (4.19b)\} \subsetneq \{y \in \{0,1\}^{|\mathcal{Y}|} \mid (4.19a)\text{-}(4.19f)\}$.

Proof. Let $S := \{y \mid (4.3a), (4.5), (4.19a), (4.19b)\}$ and $T := \{y \mid (4.3a), (4.19a)\text{-}(4.19f)\}$. We first show that $S \subseteq T$. Pick any point $y \in S$. We need to show that y satisfies

(4.19c)-(4.19f). Point y satisfies (4.19c) and (4.19d) because those are relaxations of (4.5a) and (4.5b), respectively, by the fact that $\mathcal{B}$ (set of binary operators), $\mathcal{U}$ (set of unary operators), and $\mathcal{L}$ (set of operands) are a partition of $\mathcal{O}$ (set of operators and operands). We can also show that y satisfies (4.19e) by

$$\sum_{o \in \mathcal{U} \cup \mathcal{L}} y_n^o \leq 1 - \sum_{o \in \mathcal{B}} y_n^o = 1 - \sum_{o \in \mathcal{O}} y_{2n}^o,$$

where the inequality and the equality hold by (4.19a) and (4.5b), respectively. Similarly, we can show that y satisfies (4.19f) by

$$\sum_{o \in \mathcal{L}} y_n^o \leq 1 - \sum_{o \in \mathcal{B} \cup \mathcal{U}} y_n^o = 1 - \sum_{o \in \mathcal{O}} y_{2n+1}^o,$$

where the inequality and the equality hold by (4.19a) and (4.5a), respectively. Therefore, y satisfies all the constraints in T, and consequently $S \subseteq T$ holds.

We next show that there exists $y \in T \setminus S$. Let $\mathcal{N} = [7]$, $y_1^{x_1} = y_2^{x_2} = y_7^{x_3} = 1$, and otherwise $y_n^o = 0$. Figure 4.2b shows the expression tree corresponding to y. The y satisfies (4.19a) and (4.19b). Also, y satisfies (4.19a) and (4.19b) because every left-hand-side value is zero. However, y does not satisfy (4.5) because x_2 and x_3 cannot be assigned unless an operator is assigned to their parents. Therefore, the proof is complete. □

Value-Defining Constraints

Value-defining constraints enforce that the value of $v_{i,n}$ is computed based on the solution expression tree and data points. Specifically, if an operand is assigned to node n, then $v_{i,n}$ is equal to the value of the operand. If a unary operator $\otimes(x)$ is assigned to node n, then $v_{i,n}$ is equal to $\otimes(v_{i,2n+1})$. If a binary operator $\otimes$ is assigned to node n, then $v_{i,n}$ is equal to $v_{i,2n} \otimes v_{i,2n+1}$. Cozad and Sahinidis (2018) introduce a value-defining constraint for each data point, each node, and each operator or operand, given in (4.20)–(4.29) in Section 4.7.2. All these constraints are necessary to ensure the correct prediction values for each data point given an expression tree.

We propose a set of improved value-defining constraints, (4.6) together with (4.22)–(4.29) in Section 4.7, which is taken from Cozad and Sahinidis (2018):

$$v_{i,n} \le \sum_{j=1}^{d} x_{i,j} y_n^{x_j} + v_{\text{up}} \sum_{o\in\mathcal{B}\cup\mathcal{U}\cup\{\text{cst}\}} y_n^o, \qquad \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}, \tag{4.6a}$$

$$v_{i,n} \ge \sum_{j=1}^{d} x_{i,j} y_n^{x_j} + v_{\text{lo}} \sum_{o\in\mathcal{B}\cup\mathcal{U}\cup\{\text{cst}\}} y_n^o, \qquad \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}. \tag{4.6b}$$

Our value-defining constraints replace (4.20)-(4.21) with (4.6). Both (4.6) and (4.20)-(4.21) represent the following disjunction:

$$\bigvee_{o\in\{x_1,\ldots,x_d\}} \begin{bmatrix} y_n^o = 1 \\ v_{i,n} = x_{i,j}, \\ \forall i \in [n_{\text{data}}] \end{bmatrix} \bigvee \begin{bmatrix} \sum_{o\in\mathcal{O}} y_n^o = 0 \\ v_{i,n} = 0, \\ \forall i \in [n_{\text{data}}] \end{bmatrix} \bigvee \begin{bmatrix} \sum_{o\in\mathcal{B}\cup\mathcal{U}\cup\{\text{cst}\}} y_n^o = 1 \\ v_{\text{lo}} \le v_{i,n} \le v_{\text{up}}, \\ \forall i \in [n_{\text{data}}] \end{bmatrix}, \qquad \forall n \in \mathcal{N}. \tag{4.7}$$

Note that if $\sum_{o\in\mathcal{B}\cup\mathcal{U}\cup\{\text{cst}\}} y_n^o = 1$ (i.e., node n exists), then the value of $v_{i,n}$ is determined by (4.22)–(4.29). Our formulation reduces the number of constraints. The number of constraints (4.6) is $n_{\text{data}}|\mathcal{N}|$, while the number of constraints (4.20) and (4.21) is $n_{\text{data}}|\mathcal{N}|(d+1)$. We show in Lemma 11 that our formulation does not change the feasible set and, in fact, reduces the space of the relaxation.

We first show in Lemma 10 that we can merge k big-M constraints associated with different constant bounds on an identical function. This merge can reduce the space of the relaxation while it does not change the feasible space.

Lemma 10. *Let k be a positive integer. Let $w \in \mathbb{R}^k$. Let $M \ge \max_{i\in[k]} w_i$. Let $\mathcal{F}_B = \{y \in \{0,1\}^k \mid \sum_{i=1}^k y_i = 1\}$ and $\mathcal{F}_C = \{y \in \mathbb{R}_+^k \mid \sum_{i=1}^k y_i = 1\}$. Consider sets S and T, where*

$$S := \{(z,y) \in \mathbb{R}\times\mathbb{R}^k \mid z \le \sum_{i\in[k]} w_i y_i\},$$

$$T := \{(z,y) \in \mathbb{R}\times\mathbb{R}^k \mid z \le w_i y_i + M(1-y_i),\ \forall i \in [k]\}.$$

Then, the following relations hold:

$$\{(z, y) \in S \mid y \in \mathcal{F}_B\} = \{(z, y) \in T \mid y \in \mathcal{F}_B\}, \tag{4.8a}$$

$$\{(z, y) \in S \mid y \in \mathcal{F}_C\} \subseteq \{(z, y) \in T \mid y \in \mathcal{F}_C\}. \tag{4.8b}$$

Further, the inclusion in (4.8b) is strict if $|\{i \in [k] : w_i < M\}| \geq 2$.

Proof. The proof of this result is given in Section 4.8. □

Next, we show that the new constraints do not change the feasible set of the MINLP but improve its continuous relaxation.

Lemma 11. *Let* $\mathcal{F}_B = \{(y, v) \in \{0, 1\}^{|\mathcal{Y}|} \times [v_{lo}, v_{up}]^{n_{data}|\mathcal{N}|} \mid \text{(4.19a)}\}$ *and* $\mathcal{F}_C = \{(y, v) \in [0, 1]^{|\mathcal{Y}|} \times [v_{lo}, v_{up}]^{n_{data}|\mathcal{N}|} \mid \text{(4.19a)}\}$. *It holds that*

$$\{(y, v) \in \mathcal{F}_B \mid \text{(4.6)}\} = \{(y, v) \in \mathcal{F}_B \mid \text{(4.20)}, \text{(4.21)}\}, \tag{4.9a}$$

$$\{(y, v) \in \mathcal{F}_C \mid \text{(4.6)}\} \subseteq \{(y, v) \in \mathcal{F}_C \mid \text{(4.20)}, \text{(4.21)}\}. \tag{4.9b}$$

Further, the inclusion in (4.9b) is strict if there is a data point (x_i, z_i) *such that there are three or more unique component values in* x_i*, that is, there exist at least three indices* j, k, l*, such the corresponding components of* x_i *are distinct, i.e.* $x_{i,j} \neq x_{i,k} \neq x_{i,l} \neq x_{i,j}$.

Proof. Constraints (4.6), (4.19a), (4.20), and (4.21) are all separable in $n \in \mathcal{N}$. Thus, it is sufficient to show that (4.9) hold for a specific n. We introduce $y_n^{\text{none}} := 1 - \sum_{(n',o) \in \mathcal{Y}: n'=n} y_n^o$. By definition, $y_n^{\text{none}} + \sum_{(n',o) \in \mathcal{Y}: n'=n} y_n^o = 1$. Then, we can merge all "$v_{i,n} \leq \cdots$" constraints in (4.20) and (4.21) into (4.6a) for all $n \in \mathcal{N}$, an action that corresponds to merging constraints in T to the constraint in S in Lemma 10. Similarly, we merge all "$v_{i,n} \geq \cdots$" constraints in (4.20) and (4.21) into (4.6b) for all $n \in \mathcal{N}$. By Lemma 10, this merge does not change the feasible set. Further, it strictly reduces the space of the relaxation if there is a data point, x_i, such that there are three or more distinct component values in x_i, because the values of x_i correspond to w in Lemma 10. Therefore, the proof is complete. □

Redundancy-Eliminating Constraints

In this section, we introduce constraints that remove equivalent expression trees. We say two expression trees are equivalent if the domains are identical and given every point in the domain, both functional values are same. By removing equivalent expression trees except one from the feasible space, we expect to explore fewer nodes in branch-and-bound tree. We consider three kinds of redundant operations related to association property, operations on constants, and nested operations. Table 4.1 shows an example of equivalent expressions for each redundant type.

Redundancy type	Example	Constraints
Association property	$x_1 - (x_2 - 3) = x_1 + (3 - x_2)$	(4.10a), (4.10b)
Operations on constants	$2 = \sqrt{4} = 1.5 + 0.5$	(4.10c), (4.30d)
Nested operations	$x = e^{\log(x)} = \log(e^x)$	(4.30e), (4.30f)

Table 4.1: Redundancy removing constraints.

In addition to the redundancy-eliminating constraints (4.30d)-(4.30f) from Cozad and Sahinidis (2018), we introduce the following constraints:

$$y_n^{+} + y_{2n+1}^{-} \le 1, \qquad n \notin \mathcal{N}_{\text{perfect}}, \tag{4.10a}$$

$$y_n^{*} + y_{2n+1}^{/} \le 1, \qquad n \notin \mathcal{N}_{\text{perfect}}, \tag{4.10b}$$

$$y_{2n+1}^{\text{cst}} \le y_n^{+} + y_n^{*}, \qquad n \notin \mathcal{T}, \tag{4.10c}$$

where $\mathcal{N}_{\text{perfect}}$ is a set of non-leaf nodes whose rooted subtree of $\mathcal{N}$ is a perfect binary tree.[3] Since $\mathcal{N}$ can be any arbitrary tree, we add (4.10a) and (4.10b) only for node $n \in \mathcal{N}_{\text{perfect}}$ otherwise, one of the redundant expressions can be infeasible because the set of possible tree structures is limited by $\mathcal{N}$. Specifically, the constraints exclude all equivalent expressions except the first expression of the examples in Table 4.1.

Redundancy induced by the association property is not considered in Cozad and Sahinidis (2018), while our redundancy-removing constraints (4.10a) and (4.10b) exclude such cases. For example, all the expression trees in Figures 4.3 and 4.4 are feasible in the formulation in Cozad and Sahinidis (2018); but, only (a) in Figures 4.3 and 4.4

[3] A perfect binary tree is a binary tree in which all nonterminal nodes have two children and all terminal nodes have the same depth.

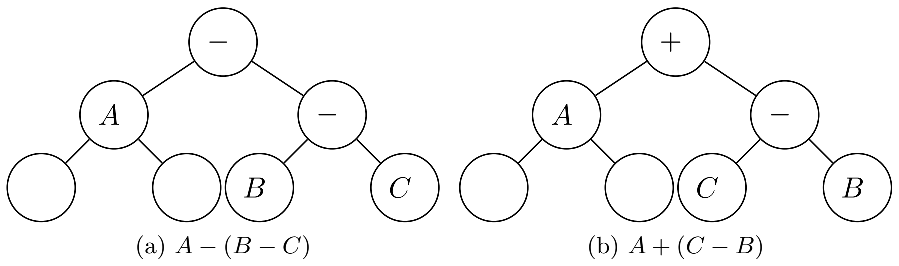


Figure 4.3: Two equivalent expression trees with addition and subtraction. Both are feasible in Cozad and Sahinidis (2018), but only (a) is feasible in (4.10a).

are feasible in our formulation.

Constraint (4.10c) allows the right child to be a constant only if $+$ or $*$ is assigned to node n. The constraints exclude $-C$ and $/C$ because equivalent expressions $+(-C)$ and $*(1/C)$ are feasible. In addition, they exclude $\sqrt{C}$, $\exp C$, and $\log C$ because we can represent them as a single constant node. This type of redundancy is considered in Cozad and Sahinidis (2018), and the corresponding constraints are (4.30a)–(4.30c). We show in Lemma 12 that the substitution (4.10c) for (4.30a)–(4.30c) results in a tighter relaxation.

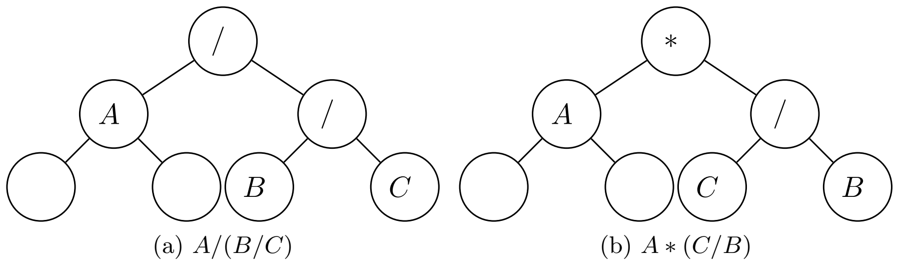


Figure 4.4: Two equivalent expression trees with multiplication and division. Both are feasible in Cozad and Sahinidis (2018), but only (a) is feasible in (4.10b).

Lemma 12. *Let $\mathcal{F}_B$ the set of binary variables y that satisfy the tree-defining constraints and let $\mathcal{F}_C$ be its continuous relaxation:*

$$\mathcal{F}_B = \{y \in \{0,1\}^{|\mathcal{Y}|} \mid (4.5), (4.19a), (4.19b)\},$$
$$\mathcal{F}_C = \{y \in [0,1]^{|\mathcal{Y}|} \mid (4.5), (4.19a), (4.19b)\}.$$

It follows that

$$\{y \in \mathcal{F}_B \mid (4.10c)\} = \{y \in \mathcal{F}_B \mid (4.30b)\text{-}(4.30d)\}, \tag{4.12a}$$

$$\{y \in \mathcal{F}_C \mid (4.10c)\} \subsetneq \{y \in \mathcal{F}_C \mid (4.30b)\text{-}(4.30d)\}. \tag{4.12b}$$

Proof. Recall (4.30b)-(4.30d):

$$\begin{aligned} y^{\text{cst}}_{2n+1} &\le 1 - \sum_{o \in \mathcal{U}} y^o_n, && n \notin \mathcal{T}, \\ y^{\text{cst}}_{2n+1} &\le 1 - y^-_n, && n \notin \mathcal{T}, \\ y^{\text{cst}}_{2n+1} &\le 1 - y^/_n, && n \notin \mathcal{T}. \end{aligned}$$

By relaxing (4.5a), we get the following inequality:

$$y^{\text{cst}}_{2n+1} \le 1 - y^{\text{none}}_n, \qquad n \notin \mathcal{T},$$

where $y^{\text{none}}_n := 1 - \sum_{o \in \mathcal{B} \cup \mathcal{U}} y^o_n$. Let us consider that those constraints are defining the upper bound of y^{cst}_{2n+1} depending on the choice of y_n. For example, $y^{\text{cst}}_{2n+1} \le 1 - y^-_n$ enforces the upper bound by 0 if $y^-_n = 1$; otherwise it relaxes this constraint. By Lemma 10, we can merge the constraints for each n. The merged constraints are

$$y^{\text{cst}}_{2n+1} \le 1 - \sum_{o \in \mathcal{U}} y^o_n - y^-_n - y^/_n - y^{\text{none}}_n = y^+_n + y^*_n, \qquad n \notin \mathcal{T}.$$

By Lemma 10, this merge does not change the feasible set; it reduces the feasible set of the relaxation. □

Implication Cuts

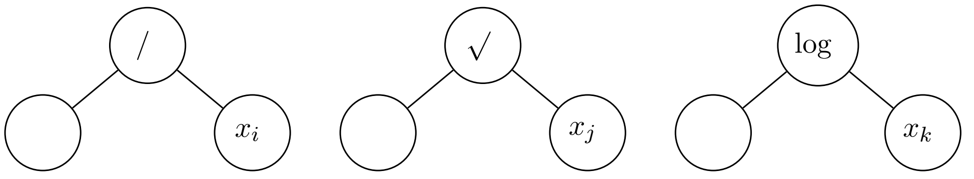
Implication cuts are motivated by the fact that some operators are domain-restricted, for example, $/$, $\sqrt{}$, and log. From a set of given data points, we can identify the characteristics of the independent variables:

$$\mathcal{X}_{\text{posi}} = \{i \in [d] \mid \exists j \in [n_{\text{data}}] : x_{i,j} > 0\}, \tag{4.13a}$$

$$\mathcal{X}_{\text{nega}} = \{i \in [d] \mid \exists j \in [n_{\text{data}}] : x_{i,j} < 0\}, \tag{4.13b}$$

$$\mathcal{X}_{\text{zero}} = \{i \in [d] \mid \exists j \in [n_{\text{data}}] : x_{i,j} = 0\}\,. \tag{4.13c}$$

All invalid expression trees with up to depth one are described in Figure 4.5. We can



(a) $(\cdot)/x_i$, $i \in \mathcal{X}_{\text{zero}}$ (b) $\sqrt{x_j}$, $j \in \mathcal{X}_{\text{nega}}$ (c) $\log(x_k)$, $k \in \mathcal{X}_{\text{nega}} \cup \mathcal{X}_{\text{zero}}$

Figure 4.5: Invalid expression trees because of domain restriction.

write the constraints that restrict the expressions in Figure 4.5 as follows:

$$y_n^{/} + y_{2n+1}^{x_j} \leq 1, \qquad \forall j \in \mathcal{X}_{\text{zero}},\ \forall n \notin \mathcal{T}, \tag{4.14a}$$
$$y_n^{\sqrt{}} + y_{2n+1}^{x_j} \leq 1, \qquad \forall j \in \mathcal{X}_{\text{nega}},\ \forall n \notin \mathcal{T}, \tag{4.14b}$$
$$y_n^{\log} + y_{2n+1}^{x_j} \leq 1, \qquad \forall j \in \mathcal{X}_{\text{nega}} \cup \mathcal{X}_{\text{zero}},\ \forall n \notin \mathcal{T}. \tag{4.14c}$$

Constraint (4.14) excludes an expression tree that includes the expressions in Figure 4.5 as a subtree. We can generate more invalid trees from depth-two or higher-depth trees. One example is $\sqrt{x_j x_k}$ for $j \in \mathcal{X}_{\text{posi}}$ and for $k \in \mathcal{X}_{\text{nega}}$. The corresponding constraint is

$$y_n^{\sqrt{}} + y_{2n+1}^{*} + y_{4n+2}^{x_j} + y_{4n+3}^{x_k} \leq 3,$$
$$\forall j \in \mathcal{X}_{\text{posi}},\ \forall k \in \mathcal{X}_{\text{nega}},\ \forall n \in \mathcal{N} : \{4n+2, 4n+3\} \subseteq \mathcal{N}. \tag{4.15}$$

Note that the expressions in Figure 4.5 are already infeasible in both our formulation and the formulation in Cozad and Sahinidis (2018) because of (4.26d), (4.27c), and (4.29c). However, adding implication cuts reduces the feasible space of the relaxation. Example 9 shows that the solution $y_n^{\sqrt{}} = y_{2n+1}^{x_j} = 0.9$ for $j \in \mathcal{X}_{\text{nega}}$ is feasible in the space of the relaxation of the formulation without implication cuts, whereas it violates (4.14b).

Example 9. *We consider a symbolic regression problem with $\mathcal{N} = \{1, 3\}$, $\mathcal{P} = \{\sqrt{}\}$ and a single data point $(x_{1,1}, z_1) = (-1, 5)$. Suppose that $v_{lo} = -10$, $v_{up} = 10$, and $\epsilon = 0.01$. To streamline the presentation, we assume that $y_1^{cst} = y_3^{cst} = 0$. The formulation without* *implication cuts is as follows:*

$$
\begin{aligned}
\min \quad & (5 - v_{1,1})^2, \\
s.t. \quad & \textit{Tree-Defining Constraints:} \\
& y_1^{\surd} + y_1^{x_1} \leq 1, \quad y_3^{x_1} \leq 1, \quad y_1^{x_1} + y_3^{x_1} \geq 1, \quad y_1^{\surd} = y_3^{x_1}, \\
& \textit{Value-Defining Constraints:} \\
& v_{1,1} \leq (-1) y_1^{x_1} + 10 y_1^{\surd}, \quad v_{1,1} \geq (-1) y_1^{x_1} - 10 y_1^{\surd}, \\
& v_{1,3} \leq (-1) y_3^{x_1}, \quad v_{1,3} \geq (-1) y_3^{x_1}, \\
& v_{1,1}^2 - v_{1,3} \leq 90(1 - y_1^{\surd}), \quad v_{1,1}^2 - v_{1,3} \geq -10(1 - y_1^{\surd}), \\
& 0.01 - v_{1,3} \leq 10.01(1 - y_1^{\surd}), \\
& \textit{Variable Restrictions:} \\
& -10 \leq v_{1,1}, v_{1,3} \leq 10, \quad y_1^{\surd}, y_1^{x_1}, y_3^{x_1} \in \{0, 1\}.
\end{aligned}
$$

Note that there is no redundancy-removing constraint because we consider only a limited set of operators and a limited set of nodes. We consider the relaxation that replaces $y_1^{\surd}, y_1^{x_1}, y_3^{x_1} \in \{0,1\}$ *with* $y_1^{\surd}, y_1^{x_1}, y_3^{x_1} \in [0,1]$. *Consider the solution* $(y_1^{\surd}, y_1^{x_1}, y_3^{x_1}, v_{1,1}, v_{1,3}) = (0.9, 0.1, 0.9, 0, -0.9)$. *The solution is feasible in the relaxation, whereas it violates* $y_1^{\surd} + y_3^{x_1} \leq 1$, *which is* (4.14b).

Symmetry-breaking Constraints

Symmetry-breaking constraints remove equivalent expression trees because of a symmetric operator including addition ($+$) and multiplication ($*$), for example, $x_1 + x_2 = x_2 + x_1$. We retain only those expression trees in which the left argument value is greater than or equal to the right argument value for the first data point:

$$v_{1,2n} - v_{1,2n+1} \geq (v_{\text{lo}} - v_{\text{up}})(1 - y_n^{+} - y_n^{*}), \qquad n \in \mathcal{N}_{\text{perfect}}. \tag{4.16}$$

Symmetry-breaking constraints are discussed in (5) in Cozad and Sahinidis (2018). We rewrite the constraints in terms of our notations because we assume that $\mathcal{N}$ corresponds to a full binary tree whereas Cozad and Sahinidis (2018) assumes that $\mathcal{N}$ corresponds to a perfect binary tree.

4.3.6 Summary

We summarize the formulation and our contributions in Table 4.2. We separate the formulation by rows, *i.e.*, the objective function and constraints. The third column 'Convexify' describes the type of functions used in the objective and the constraints where the function of a constraint $f(y, c, v) \leq 0$ is $f(y, c, v)$. The fourth column 'remove additional equivalent expressions' express whether the constraints remove redundant expression trees not considered in Cozad and Sahinidis (2018) or not. The fifth column 'reduce relaxation space' expresses whether the constraints reduces the LP relaxation space compared to the formulation in Cozad and Sahinidis (2018) or not.

Table 4.2: Summary of the formulation and contributions compared with the benchmark formulation Cozad and Sahinidis (2018).

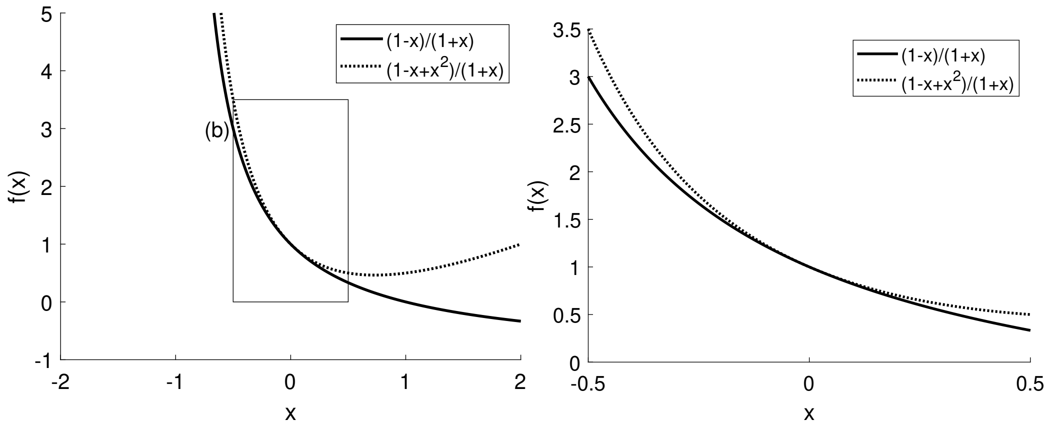
Categories	Variables	Convexity	Remove additional equiv. expressions	Reduce relaxation space
Objective	v	convex		
Tree-defining constraints	y, c	linear	✓	✓
Value-defining constraints	y, c, v	nonconvex		✓
Redundancy-eliminating constraints	y	linear	✓	✓
Implication cuts	y	linear		✓
Symmetry-breaking constraints	y, v	linear		

The formulation we propose for symbolic regression is

$$\begin{aligned}
\min_{y,c,v} \quad & \frac{1}{n_{\text{data}}} \sum_{i=1}^{n_{\text{data}}} (z_i - v_{i,1})^2 \\
\text{s.t.} \quad & (4.5), (4.19a)\text{-}(4.19b), && \text{(Tree-defining constraints)} \\
& (4.6), (4.22)\text{-}(4.29), && \text{(Value-defining constraints)} \\
& (4.10), (4.30d)\text{-}(4.30f), && \text{(Redundancy-eliminating constraints)} \\
& (4.14), && \text{(Implication cuts)} \\
& (4.16), && \text{(Symmetry-breaking constraints)} \\
& y_n^o \in \{0, 1\}, && \forall (n, o) \in \mathcal{Y}, \\
& c_{\text{lo}} \leq c_n \leq c_{\text{up}}, && \forall n \in \mathcal{N}, \\
& v_{\text{lo}} \leq v_{i,n} \leq v_{\text{up}}, && \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}.
\end{aligned}$$

4.4 Sequential Tree Construction Heuristic

In this section, we propose a heuristic that repeatedly searches from a given solution to build an expression tree corresponding to a more comprehensive formula with a lower training error. It is motivated by the fact that a comprehensive formula can be approximated by a simple formula, and we observe that those two formulas have similar structures. For example, $\frac{1-x+x^2}{1+x}$ can be approximated by $\frac{1-x}{1+x}$ when $|x|$ is small, see Figure 4.6. We can achieve the comprehensive formula by adding x^2 to the simple formula. Another example can be found in Kepler's third law, where the comprehensive formula $d = \sqrt[3]{c\tau^2(M+m)}$ is approximated by a simple formula $d = \sqrt[3]{c\tau^2 M}$ for $M \gg m$.



(a) $x \in (-1, 2)$ (b) $x \in (-0.5, 0.5)$

Figure 4.6: Illustration of $\frac{1-x+x^2}{1+x}$ and $\frac{1-x}{1+x}$.

Our approach is motivated by a heuristic framework for general MINLPs that searches an improved solution from the neighbors of the current solution, namely, local branching proposed by Fischetti and Lodi (2003). A local branching heuristic iteratively explores the neighbors by solving a restricted MINLP with a branch-and-bound solver. We develop a *sequential tree construction heuristic (STreCH)* based on the formulation in Section 4.3. In Section 4.4.1 we define the neighbors of an expression tree that is the core of our heuristic, and in Section 4.4.2 we discuss how to speed up the heuristic.

4.4.1 Definition of Neighbors and a Basic Heuristic

We define the distance between two expression trees as the number of nodes assigned different operators/operands. There are three cases of a node with different assignment:

- a change of a node assignment (from an operator/operand to another operator/operand),
- an addition of a new node, and
- a deletion of a node.

We consider constant as an operand. This distance does not count a change from a constant value to another constant value. For example, the distance between two trees in Figure 4.7 is three because of one change (from x_1 to $*$) and two additions of a node. The change in the constant value (from 2.0 to 1.5) is not counted.

We define a k-neighbor of a given solution $\bar{y}$ as an expression tree with distance at most k. Let $\mathcal{N}_{\text{active}}(\bar{y})$ be the set of nodes in which an operator/operand is assigned. Let $o(n, \bar{y})$ for $n \in \mathcal{N}_{\text{active}}(\bar{y})$ be the operator/operand that is assigned to node n. The set of k-neighbors of $\bar{y}$, $\mathcal{NB}_k(\bar{y})$, is represented as follows:

$$\delta(\bar{y}, y) = \sum_{n \in \mathcal{N}_{\text{active}}(\bar{y})} (1 - y_n^{o(n,\bar{y})}) + \sum_{n \notin \mathcal{N}_{\text{active}}(\bar{y})} \sum_{o \in \mathcal{O}} y_n^o, \tag{4.17}$$

$$\mathcal{NB}_k(\bar{y}) = \{ y \mid \delta(\bar{y}, y) \leq k \}. \tag{4.18}$$

The restricted MINLP that searches k-neighbors needs a single additional linear constraint described in (4.18).

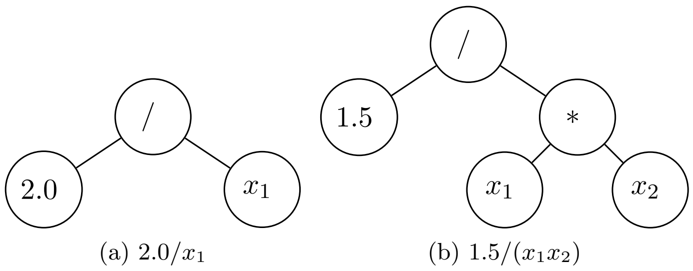


Figure 4.7: Two expression trees with distance three.

Our basic heuristic works as follows. It first obtains an initial solution by solving a small-sized problem. At each iteration, it searches for an improvement of the current solution by exploring its neighbors by solving a restricted MINLP with (4.18). It terminates by optimality (the objective value is less than a given tolerance ϵ) or by time limit. In our implementation we encounter the following situations:

- When the tree size (the number of nodes) of the incumbent solution is large, it takes a long time to search its neighbors.
- There is no better solution in its neighbor at some iteration.

We next propose some ideas to mitigate these situations.

Algorithm 2: `ResMINLP` (approximately solving a restricted MINLP).

```
Data: $(x, z)$, $\mathcal{P}$, $\mathcal{N}$; (optional) $\bar{y}$, $k_1$, $k_2$, $\beta$, $\gamma$
Result: An expression tree $(y, c)$ found by MINLP and its training error $\omega$
1 Formulate a MINLP with with data $(x, z)$, operators $\mathcal{P}$, nodes $\mathcal{N}$;
2 if $\bar{y}$, $k_1$, and $k_2$ are given then
3     Add constraint $k_1 \leq \delta(\bar{y}, y) \leq k_2$ to search within $\mathcal{NB}_{k_2}(\bar{y}) \setminus \mathcal{NB}_{k_1}(\bar{y})$;
4 if $\bar{y}$ and $\beta$ are given then
5     Fix some variables in $y$ by given solution $\bar{y}$ and fix level $\beta$;
6 if $\gamma$ is given then
7     Set the node limit of a branch-and-bound solver to $\gamma$;
8 Solve the problem with a branch-and-bound solver;
9 return $(y, c, \omega)$;
```

4.4.2 Heuristics to Speed Up Ideas and STreCH

In this section we propose STreCH, whose pseudocode is described in Algorithm 4. Algorithm 2 describes a restricted MINLP solve, and Algorithm 3 describes the solution improvement procedure. In addition, we propose a number of heuristics to speed up the solution process that are described next:

Early Termination A branch-and-bound solver returns an optimal solution with a proof of optimality. However, proving optimality for a restricted problem does not

Algorithm 3: `Improve` (procedure to find an improved solution from a solution).

```
   Data: $(x, z)$, $\mathcal{P}$, $\mathcal{N}$, solution $(\bar{y}, \bar{c}, \bar{\omega})$
   Result: An improved solution if found, otherwise the given solution
 1 $(k_1, k_2, \beta, \gamma) \leftarrow (0, k_{\text{init}}, \beta_{\text{init}}, \gamma_{\text{init}})$ ;          // Initialize the parameters.
 2 repeat                                          // Repeatedly search neighbors.
 3     $(y, c, \omega) \leftarrow$ ResMINLP$(x, z, \mathcal{P}, \mathcal{N}, k_1, k_2, \beta, \gamma)$;
 4     if $\omega < \bar{\omega}$ then
 5         return $(y, c, \omega)$ ;                        // Return an improved solution.
 6     else if Terminated by node limit and $10\gamma \leq \gamma_{\max}$ then
 7         $\gamma \leftarrow 10\gamma$ ;                         // Spend more time on this problem.
 8     else                                                    // Spent enough time.
 9         $(k_1, k_2) \leftarrow (k_2, k_2 + 2)$ ;                   // Change the neighbor set.
10         if $k_2 > k_{\max}$ then
11             $(k_1, k_2) \leftarrow (0, k_{\text{init}})$;
12             $\beta \leftarrow \beta + 1$;
13             if $\beta > \beta_{\max}$ then
14                 return $(\bar{y}, \bar{c}, \bar{\omega})$ ;            // Return the given solution.
15 until Time limit reached;
```

guarantee optimality of the whole problem. Therefore, we terminate the solver under one of the following conditions: (1) when it finds an improved solution, or (2) it reaches a time-limit to prove optimality. First, we stop an iteration when the solver finds a solution whose training error is smaller than $(100 * \delta)\%$ of the training error of the incumbent. Second, we stop an iteration when it reaches a predetermined time limit or node limit.

Start Value We provide the incumbent as a starting point. In general, it is not always efficient especially when we are looking for an optimal solution with a proof of optimality. However, it helps in combination with early termination.

Fix a Part of an Expression Tree As the size of an expression tree grows, the size of k-neighbors increases. We fix the top part of an expression tree to reduce the search space. Specifically, given an incumbent and $\beta \in \mathbb{Z}_{++}$, we fix all nodes that have at least one descendant of distance β. For example, when $\beta = 1$, we fix all nonleaf nodes. When $\beta = 2$, we fix all nodes that have a grandchild node.

With the implementation of early termination, there are three situations when a MINLP solver terminates at an iteration. Let $\bar{y}$ denote the current incumbent and k denote the current distance. We define the next iteration for each situation as follows:

1. The solver returns a better solution. Then, we solve a MINLP restricted by the neighbors of the returned solution.

2. The solver proves that there is no better solution within the neighbors. Then, we search in a larger neighborhood, $\mathcal{NB}_{k'}(\bar{y}) \setminus \mathcal{NB}_k(\bar{y})$ where $k' > k$.

3. The solver terminates by time limit or node limit and no better solution has been found. Then, we increase the time limit or the node limit in the next iteration.

The pseudocode of STreCH is described in Algorithms 2 to 4. Algorithm 2 describes a restricted MINLP. Algorithm 3 describes the solution-improving procedure. The inputs of Algorithm 3 are the solution $\bar{y}$ and the parameters that are given in the beginning and not changed during the procedure: k_{init} is the initial distance to define neighbors, $k_{\max}$ is the maximum distance of candidate neighbors, β_{init} is the initial node fix level, $\beta_{\max}$ is the maximum node fix level, γ_{init} is the initial node limit for the branch-and-bound tree, and $\gamma_{\max}$ is the maximum node limit for the branch-and-bound tree. Algorithm 4 describes the overall loop.

We recommend solving a single MINLP by limiting the number of nodes in the branch-and-bound tree instead of limiting time in the heuristic. The reason is that limiting the number of node guarantees that the same solution will be reproduced at each iteration whereas limiting time may return a different solution depending on how much resource is available on the computing machine.

4.5 Computational Experiments

We perform computational experiments on the improved formulation and the sequential tree construction heuristic. The first experiment tests our ability to find a global solution. We investigate whether the new constraints deliver an improvement in computation. The second experiment tests the ability to find a good approximated symbolic function with limited information. We assume that a limited number of observations is

Algorithm 4: STreCH.

```
  Data: $(x, z)$, $\mathcal{P}$, $\mathcal{N}$, $\mathcal{N}_{\text{init}}$ (the node set for the initial problem)
  Result: An expression tree $(y, v)$ and its training error $\omega$
1 $(y, c, \omega) \leftarrow$ ResMINLP$(x, z, \mathcal{P}, \mathcal{N}_{\text{init}})$ ;        // Solve an initial problem.
2 repeat                                          // Repeatedly improve a solution.
3     $(y', c', \omega') \leftarrow$ Improve$(x, z, \mathcal{P}, \mathcal{N}, y, c, \omega)$;
4     if $\omega' < \omega$ then
5         $(y, c, \omega) \leftarrow (y', c', \omega')$ ;                // Update the incumbent.
6     else
7         break;
8 until Time limit reached;
9 return $(y, c, \omega)$;
```

given and no additional information such as the unit of variables is available. We compare our methods with AI Feynman (Udrescu and Tegmark, 2020; Udrescu et al., 2020), which is a state-of-the-art symbolic regression solver specialized for physics formulas.

Test Problems We test 71 formulas from the Feynman database for symbolic regression (Udrescu and Tegmark, 2020) whose operator set is a subset of $\{+, -, *, /, \sqrt{}\}$. Table 4.3 shows that all formulas can be represented by an expression tree of depth five. We assume that (i) no unit information is available, (ii) we have a small number of observations (10 data points), and (iii) the observations are noisy. Although (i)–(iii) were discussed in Udrescu and Tegmark (2020) independently, the combination of all three was not discussed.

Depth	1	2	3	4	5	Total
# of formulas	4	25	22	7	13	71

Table 4.3: Distribution of the required depth to represent a formula in the test problems.

Hardware and Software Our computational experiments are performed on a computer with Intel Xeon Gold 6130 CPU cores running at 2.10 GHz and 192 GB of memory. The operating system is Linux Ubuntu 18.04. The code is written in Julia 1.5.3

with SCIP 7.0.0 (Gleixner et al., 2018) as a MINLP solver that showed the best performance among open-source global MINLP solvers for this problem (Cozad and Sahinidis, 2018). The code is available at `https://github.com/jongeunkim/STreCH`.

4.5.1 Comparison of MINLP Formulations

We compare the formulations described in Section 4.3 with the formulation from Cozad and Sahinidis (2018).

Experimental Setup for Comparing MINLP Formulations

We start by investigating the effect of the new optional constraints that do not need to be included in the formulation but can reduce the search space. In the experiments in Cozad and Sahinidis (2018), the variance of the computational performance is large with regard to the inclusion/exclusion of optional constraints, and the authors suggest running all possible formulations in parallel. In this experiment, we consider four formulations for each method (ours and Cozad and Sahinidis (2018)) by adding or not adding redundancy-eliminating constraints and symmetry-breaking constraints. *Imp* and *Coz* stand for the improved formulation and the formulation from Cozad and Sahinidis (2018), respectively. *-F* refers to the full formulation (adding all the constraints). *-N* refers to the formulation with only the necessary constraints (tree and value defining constraints). *-R* refers to the formulation with the necessary constraints and the redundancy-eliminating constraints. *-S* refers to the formulation with the necessary constraints and the symmetry-breaking constraints. The configuration of the formulations are described in Table 4.4. We do not consider implication cuts because all the indepen-

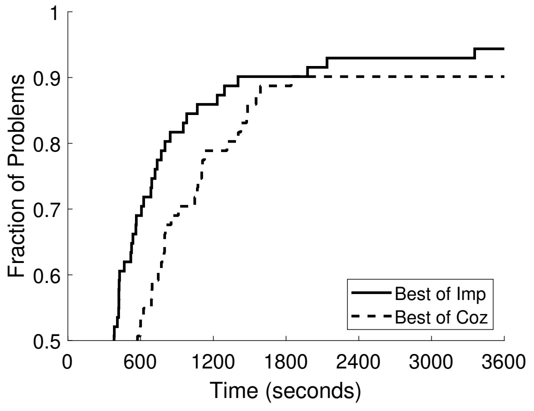
Formulations	Imp-F	Imp-R	Imp-S	Imp-N	Coz-F	Coz-R	Coz-S	Coz-N
Objective	(4.4)							
Tree	(4.5), (4.19a)-(4.19b)				(4.19)			
Value	(4.6), (4.22)-(4.29)				(4.20)-(4.29)			
Redundancy	(4.10), (4.30d)-(4.30f)		-		(4.30)		-	
Symmetry	(4.16)	-	(4.16)	-	(4.16)	-	(4.16)	-

Table 4.4: Formulations used in the experiments.

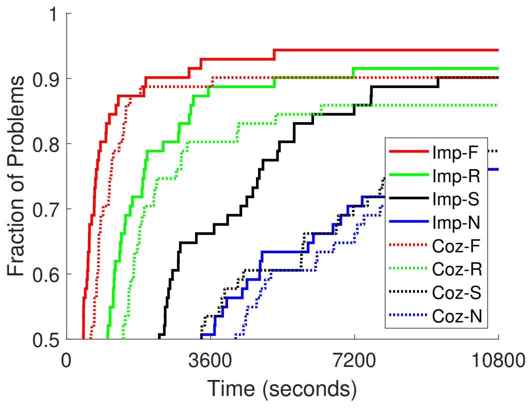
dent variables used in the test functions are positive, which means that there are no implication cuts. We limit the depth of the expression tree to two (seven nodes) in order to find an optimal solution for all instances within the prespecified time limit (three hours).

Results Comparing MINLP Formulations

First, we compare our best results with the best results of Cozad and Sahinidis (2018) in Figure 4.8a. We collect the smallest solution times among four formulations for each method. We visualize our results using performance profiles (Dolan and Moré, 2002) in Figure 4.8. Figure 4.8a shows that our formulations can solve 70% of instances within ten minutes while Cozad and Sahinidis (2018)'s can solve 50% of instances. Our formulations failed to solve 2.8% of instances (2 of 71) within three hours while Cozad and Sahinidis (2018)'s failed to solve 5.6% of instances (4 of 71) within the time limit.



(a) Best of four.



(b) All.

Figure 4.8: Comparison of solution times of MINLP solves with the improved formulations and those of Cozad and Sahinidis (2018).

Next, we compare all eight formulations in Figure 4.8b. Now we draw eight lines for each formulation. Figure 4.8b shows that our formulation with all optional constraints performs best. The formulation terminates first in more than half of the instances (36 of 71) and is in the top three in 81.7% of the instances (58 of 71). Figure 4.9 shows that the newly proposed optional constraints in the improved formulation also reduce the number of nodes in the branch-and-bound tree (BnBnodes) compared with the existing ones.

Our experiments show that it clearly is better to add all optional constraints to reduce the search space. This conclusion differs from the result in Cozad and Sahinidis (2018). We believe that this difference may be due to a difference in the branch-and-bound solvers: we use SCIP while Cozad and Sahinidis (2018) use BARON (Sahinidis, 2017).

4.5.2 STreCH and AI Feynman

The goal of this experiment is to compare the performance of our methods with the state-of-the-art symbolic regression method AI Feynman.

Experimental Setup for Heuristic Comparison

We test both methods with noisy data and perform cross-validation to select the final symbolic expression. We first generate a training set with noise and a validation and testing set without noise. For each method, we find a set of candidate formulas using the training set. Next, we select the formula from the candidates that has the lowest validation error. We then compute the testing error of the selected formula.

We consider three approaches: a STreCH-based approach, a MINLP-based approach, and AI Feynman. The first two approaches solve multiple instances with different parameter setups in parallel to collect candidate formulas. The set of parameters includes the type of formulation (adding or not adding optional constraints), the maximum depth of the expression tree, the type of constant (integer or fractional), and the bounds on the constants. The maximum depth of the expression is bounded above by five, because every formula can be expressed by an expression tree of depth five. When a single instance is solved, the STreCH-based approach uses the heuristic described in Section 4.4, and the MINLP-based approach solves the MINLP problem in Section 4.3.

We run AI Feynman ourselves because there are no reported computational experiments in our setting (running without no unit information, for a small number of observations, and noisy data). We download the AI Feynman code from `https://github.com/SJ001/AI-Feynman`. We use the default parameters except the set of operators used in brute-force because the default set does not include all operators used in tested functions. Instead, we use the largest operator set that includes all used operators. Note that the AI Feynman code itself manages computing resources in parallel. We set the time limit to one hour for each approach.

Results for Comparison of Heuristics

First, we investigate how many formulas can be rediscovered by each method. We run all methods on the dataset with ten training data points, a noise level of 10^{-4}, and no

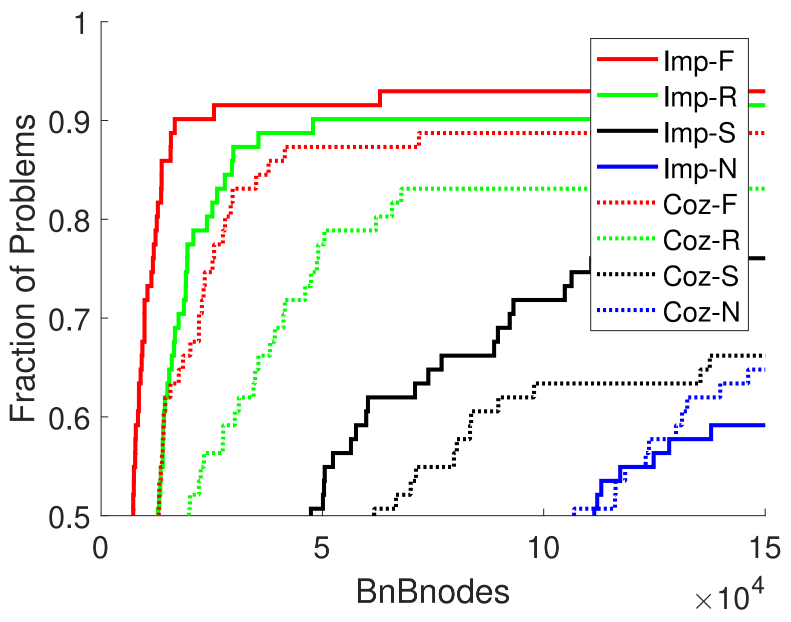


Figure 4.9: Comparison of the number of nodes in the branch-and-bound tree of MINLP solves with the improved formulations and the the formulations of Cozad and Sahinidis (2018).

unit information. In Table 4.5, we observe that every method can discover the correct formula when it can be represented by an expression tree of depth one or two. When the depth is three, the STreCH and the MINLP approaches discover twice as many formulas as AI Feynman. When the depth is four or five, the STreCH and the MINLP approaches cannot discover the original formulas, while AI Feynman discovers two formulas.

Table 4.5: Required depth to represent a formula.

Depth	# Formulas	Discovery rate (%)		
		STreCH	MINLP	AI Feynman
≤ 2	29	100.0	100.0	100.0
3	22	59.1	54.5	27.2
≥ 4	20	0.0	0.0	10.0

Next, we investigate formulas for which the methods return different solutions. We first consider formulas that were discovered by the STreCH and the MINLP approaches but not by AI Feynman. These include $\frac{q_1 q_2 r}{4\pi\epsilon r^3}$ (Feynman Eq. I.12.2) and $\frac{2I}{4\pi\epsilon c^2 r}$ (Feynman Eq. II.13.17). We suspect that this difference happens because the STreCH and the

MINLP approaches can assign any constant value at a node in the expression tree whereas AI Feynman relies on the user-specified particular constants. Specifically, when solving the problem of $\frac{q_1 q_2 r}{4\pi\epsilon r^3}$, the STreCH and the MINLP approaches can assign $0.159 (= \frac{1}{2\pi})$ at a node while AI Feynman needs a few steps in combination with a prespecified constant (π) and operators ($x \to 2x$ and $x \to \frac{1}{x}$). These few steps might hinder the ability of the path to rediscover the original formula. This weakness also has been pointed out in Austel et al. (2020).

Second, we investigate formulas that were discovered by AI Feynman but not by STreCH and the MINLP approaches. These include $\frac{m_0}{\sqrt{1-v^2/c^2}}$ and $\frac{\rho_{c_0}}{\sqrt{1-v^2/c^2}}$. AI Feynman benefits from the use of trigonometrical functions, $\arcsin(\cos(x))$, which are equivalent to $\sqrt{1-x^2}$. Third, we consider formulas where the STreCH approach could find the original formula but the MINLP approach failed. These include $\frac{(h/(2\pi))^2}{2E_n d^2}$ (Feynman Eq. III.15.14). Table 4.6 shows how the STreCH develops a solution at each iteration. Because the formula is a monomial, there are multiple sequences to reach the correct formula. For example, the correct formula can be obtained from $\frac{h^2}{E_n d^2}$, $\frac{Ch^2}{d^2}$, and $\frac{Ch}{E_n d^2}$, where $C = (8\pi^2)^{-1}$. Therefore, there are formula structures such as a monomial that the STreCH performs well.

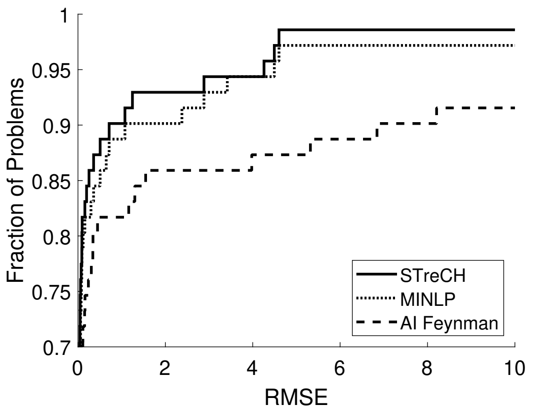
Table 4.6: Progress of STreCH to discover $\frac{(h/(2\pi))^2}{2E_n d^2}$.

Iteration	Incumbent*	Update	Time Spent (s)
1	$\frac{c_1 h}{d}$	initial solution**	75.33
2	$\frac{c_2 h}{E_n d}$	$h \to h/E_n$	2.82
3	$\frac{c_3 h}{E_n d^2}$	$d \to d^2$	68.39
4	$\frac{c_4 h}{E_n d^2}$	change the constant value	1.73
5	$\frac{c_5 h^2}{E_n d^2}$	$h \to h^2$	26.24
6	$\frac{c_6 h^2}{E_n d^2}$	change the constant value	161.77

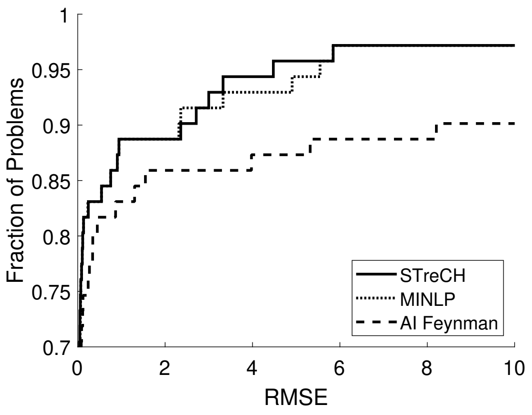
* c_1-c_6 are constant values

** achieved by solving a depth-two problem

We next compare the testing errors of the three methods. We perform the computational experiments on both noiseless and noisy data. Figure 4.10 shows the distributions of the root mean square testing errors of the results. We see that the solutions generated by STreCH and the MINLP approaches have a lower testing error compared with AI Feynman's solutions.



(a) Noiseless.



(b) Noise level of 0.01%.

Figure 4.10: Comparison of the testing errors achieved by the STreCH approach, the MINLP approach, and AI Feynman.

4.6 Conclusion

In this chapter we present MINLP-based methods for symbolic regression. We propose an improved MINLP formulation. We also propose a new heuristic, STreCH, which is based on the tighter formulation and builds an expression tree by repeatedly modifying a solution expression tree. Compared with state-of-the-art methods, our methods are able to discover more correct formulas when there is a lack of data. When an original formula is difficult to rediscover, our methods return a formula that has a lower testing error.

For future work, our method can be integrated with AI Feynman. AI Feynman decomposes the given problem into small problems and solves those using polynomial fit and brute-force methods. Since our method is good at finding a relatively simple symbolic expression, it would be a good option for solving small subproblems that arise within the AI Feynman decomposition within a tight time limit, of, say, less that a minute.

4.7 Supplements: the MINLP Formulation Proposed in Cozad and Sahinidis (2018)

The formulation proposed by Cozad and Sahinidis (2018) is written in terms of our notations.

4.7.1 Tree-Defining Constraints

The constraints are (1d)–(1i) in Cozad and Sahinidis (2018).

$$\sum_{o \in \mathcal{O}} y_n^o \le 1, \qquad n \in \mathcal{N}, \tag{4.19a}$$

$$\sum_{n \in \mathcal{N}} \sum_{j=1}^{d} y_n^{x_j} \ge 1, \tag{4.19b}$$

$$\sum_{o \in \mathcal{B} \cup \mathcal{U}} y_n^o \le \sum_{o \in \mathcal{O}} y_{2n+1}^o, \qquad n \notin \mathcal{T}, \tag{4.19c}$$

$$\sum_{o \in \mathcal{B}} y_n^o \le \sum_{o \in \mathcal{O}} y_{2n}^o, \qquad n \notin \mathcal{T}, \tag{4.19d}$$

$$\sum_{o \in \mathcal{U} \cup \mathcal{L}} y_n^o \le 1 - \sum_{o \in \mathcal{O}} y_{2n}^o, \qquad n \notin \mathcal{T}, \tag{4.19e}$$

$$\sum_{o \in \mathcal{L}} y_n^o \le 1 - \sum_{o \in \mathcal{O}} y_{2n+1}^o, \qquad n \notin \mathcal{T}. \tag{4.19f}$$

4.7.2 Value-Defining Constraints

The constraints are (1b) and (1c) in Cozad and Sahinidis (2018).

No Assignment

$$v_{i,n} \le v_{\text{up}}(1 - \sum_{o \in \mathcal{O}} y_n^o), \qquad \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}, \tag{4.20a}$$

$$v_{i,n} \ge v_{\text{lo}}(1 - \sum_{o \in \mathcal{O}} y_n^o), \qquad \forall i \in [n_{\text{data}}],\ \forall n \in \mathcal{N}. \tag{4.20b}$$

Independent Variables

$$v_{i,n} \leq x_{i,j} y_n^{x_j} + v_{\mathrm{up}}(1 - y_n^{x_j}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \in \mathcal{N},\ \forall j \in [d], \tag{4.21a}$$

$$v_{i,n} \geq x_{i,j} y_n^{x_j} + v_{\mathrm{lo}}(1 - y_n^{x_j}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \in \mathcal{N},\ \forall j \in [d]. \tag{4.21b}$$

Constant

$$v_{i,n} - c_n \leq (v_{\mathrm{up}} - c_{\mathrm{lo}})(1 - y_n^{\mathrm{cst}}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \in \mathcal{N}, \tag{4.22a}$$

$$v_{i,n} - c_n \geq (v_{\mathrm{lo}} - c_{\mathrm{up}})(1 - y_n^{\mathrm{cst}}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \in \mathcal{N}. \tag{4.22b}$$

Addition

$$v_{i,n} - (v_{i,2n} + v_{i,2n+1}) \leq (v_{\mathrm{up}} - 2v_{\mathrm{lo}})(1 - y_n^{+}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}, \tag{4.23a}$$

$$v_{i,n} - (v_{i,2n} + v_{i,2n+1}) \geq (v_{\mathrm{lo}} - 2v_{\mathrm{up}})(1 - y_n^{+}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}. \tag{4.23b}$$

Subtraction

$$v_{i,n} - (v_{i,2n} - v_{i,2n+1}) \leq (2v_{\mathrm{up}} - v_{\mathrm{lo}})(1 - y_n^{-}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}, \tag{4.24a}$$

$$v_{i,n} - (v_{i,2n} - v_{i,2n+1}) \geq (2v_{\mathrm{lo}} - v_{\mathrm{up}})(1 - y_n^{-}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}. \tag{4.24b}$$

Multiplication

$$v_{i,n} - v_{i,2n} v_{i,2n+1} \leq (v_{\mathrm{up}} - \min\{v_{\mathrm{lo}}^2, v_{\mathrm{lo}} v_{\mathrm{up}}, v_{\mathrm{up}}^2\})(1 - y_n^{*}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}, \tag{4.25a}$$

$$v_{i,n} - v_{i,2n} v_{i,2n+1} \geq (v_{\mathrm{lo}} - \max\{v_{\mathrm{lo}}^2, v_{\mathrm{up}}^2\})(1 - y_n^{*}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}. \tag{4.25b}$$

Division

$$v_{i,n} v_{i,2n+1} - v_{i,2n} \leq (\max\{v_{\mathrm{lo}}^2, v_{\mathrm{up}}^2\} - v_{\mathrm{lo}})(1 - y_n^{/}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}, \tag{4.26a}$$

$$v_{i,n} v_{i,2n+1} - v_{i,2n} \geq (\min\{v_{\mathrm{lo}}^2, v_{\mathrm{lo}} v_{\mathrm{up}}, v_{\mathrm{up}}^2\} - v_{\mathrm{up}})(1 - y_n^{/}), \qquad \forall i \in [n_{\mathrm{data}}],\ \forall n \notin \mathcal{T}, \tag{4.26b}$$

$$\epsilon y_n^{/} \leq v_{i,2n}^2, \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.26c}$$

$$\epsilon y_n^{/} \leq v_{i,2n+1}^2, \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}. \tag{4.26d}$$

Square Root

$$v_{i,n}^2 - v_{i,2n+1} \leq (\max\{v_{\text{lo}}^2, v_{\text{up}}^2\} - v_{\text{lo}})(1 - y_n^{\surd}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.27a}$$

$$v_{i,n}^2 - v_{i,2n+1} \geq (-v_{\text{up}})(1 - y_n^{\surd}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.27b}$$

$$\epsilon - v_{i,2n+1} \leq (\epsilon - v_{\text{lo}})(1 - y_n^{\surd}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}. \tag{4.27c}$$

Exponential

$$v_{i,n} - \exp(v_{i,2n+1}) \leq v_{\text{up}}(1 - y_n^{\text{exp}}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.28a}$$

$$v_{i,n} - \exp(v_{i,2n+1}) \geq (v_{\text{lo}} - \exp(v_{\text{up}}))(1 - y_n^{\text{exp}}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}. \tag{4.28b}$$

Logarithm

$$\exp(v_{i,n}) - v_{i,2n+1} \leq (\exp(v_{\text{up}} - v_{\text{lo}})(1 - y_n^{\log}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.29a}$$

$$\exp(v_{i,n}) - v_{i,2n+1} \geq (-v_{\text{up}})(1 - y_n^{\log}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}, \tag{4.29b}$$

$$\epsilon - v_{i,2n+1} \leq (\epsilon - v_{\text{lo}})(1 - y_n^{\log}), \qquad \forall i \in [n_{\text{data}}],\ \forall n \notin \mathcal{T}. \tag{4.29c}$$

4.7.3 Redundancy-Eliminating Constraints

These constraints are (2a), (2b), (3a), (3b), (4a), and (4b) in Cozad and Sahinidis (2018). Define $\mathcal{O}_{\text{pair}}$ as the set of all inverse unary operation pairs o and o' such as $(\exp, \log)$, and $((\cdot)^2, \surd)$.

$$y_{2n+1}^{\text{cst}} + \sum_{o \in \mathcal{U}} y_n^o \leq 1, \qquad n \notin \mathcal{T}, \tag{4.30a}$$

$$y_{2n+1}^{\text{cst}} + y_n^{-} \leq 1, \qquad n \notin \mathcal{T}, \tag{4.30b}$$

$$y_{2n+1}^{\text{cst}} + y_n^{/} \leq 1, \qquad n \notin \mathcal{T}, \tag{4.30c}$$

$$y_{2n}^{\text{cst}} + y_{2n+1}^{\text{cst}} \leq 1, \qquad n \notin \mathcal{T}, \tag{4.30d}$$

$$y_n^o + y_{2n+1}^{o'} \leq 1, \qquad n \notin \mathcal{T},\ (o,o') \in \mathcal{O}_{\text{pair}}, \tag{4.30e}$$

$$y_n^{o'} + y_{2n+1}^{o} \leq 1, \qquad n \notin \mathcal{T},\ (o,o') \in \mathcal{O}_{\text{pair}}. \tag{4.30f}$$

4.7.4 Symmetry-Breaking Constraints

This constraint is (5) in Cozad and Sahinidis (2018).

$$v_{1,2n} - v_{1,2n+1} \geq (v_{\text{lo}} - v_{\text{up}})(1 - y_n^+ - y_n^*), \qquad n \in \mathcal{N}_{\text{perfect}} \tag{4.31}$$

4.8 Supplements: Proof of Lemma 10

Proof. We first show (4.8a). Note that $\mathcal{F}_B = \{e_i,\ \forall i \in [k]\}$, where $e_i \in \{0,1\}^k$ is the i-th principal vector. Equality (4.8a) can be shown by

$$\{(z,y) \in S \mid y \in \mathcal{F}_B\} = \bigcup_{i=1}^{k} (S \cap \{(z,y) \mid y = e_i\}) = \bigcup_{i=1}^{k} (T \cap \{(z,y) \mid y = e_i\}) = \{(z,y) \in T \mid y \in \mathcal{F}_B\}.$$

The equality in the middle holds because of the following observations:

$$S \cap \{(z,y) \mid y = e_i\} = \begin{cases} \{(z,e_i) \mid z \leq w_i\} & \text{if } i \in [k], \\ \{(z,e_i) \mid z \leq M\} & \text{otherwise,} \end{cases}$$

$$T \cap \{(z,y) \mid y = e_i\} = \begin{cases} \{(z,e_i) \mid z \leq \min\{w_i, M\} = w_i\} & \text{if } i \in [k], \\ \{(z,e_i) \mid z \leq M\} & \text{otherwise.} \end{cases}$$

We next show (4.8b). Let $S_C := \{(z,y) \in S \mid y \in \mathcal{F}_C\}$ and $T_C := \{(z,y) \in T \mid y \in \mathcal{F}_C\}$. It holds that $S_C \subseteq T_C$ because the constraint in S dominates every constraint in T. It is sufficient to show that

$$\begin{aligned} \sum_{i\in[k]} w_i y_i + M(1 - \sum_{i\in[k]} y_i) &= w_j y_j + \sum_{i\in[k]:i\neq j} w_i y_i + M(1 - \sum_{i\in[k]} y_i) \\ &\leq w_j y_j + \sum_{i\in[k]:i\neq j} M y_i + M(1 - \sum_{i\in[k]} y_i) = w_j y_j + M(1 - y_j), \end{aligned}$$

for all $j \in [k]$.

Further, we show that the inclusion in (4.8b) becomes strict if $|\{i \in [k] : w_i < M\}| \geq 2$. Without loss of generality, assume that $w_1 \leq w_2 \leq \ldots \leq w_k$. By the assumption, $w_2 < M$. We show that there exists $(z, y) \in T_C \setminus S_C$. Consider $(\bar{z}, \bar{y})$ with $\bar{z} = \frac{1}{k}w_1 + \frac{k-1}{k}M$ and $\bar{y}_i = \frac{1}{k}$ for all $i \in [k]$. Point $\bar{y}$ is in $\mathcal{F}_C$. Point $(\bar{z}, \bar{y})$ is not in S because

$$\bar{z} - \sum_{i \in [k]} w_i \bar{y}_i = \left(\frac{1}{k}w_1 + \frac{k-1}{k}M\right) - \frac{\sum_{i \in [k]} w_i}{k} = \frac{\sum_{i=2}^{k}(M - w_i)}{k} > 0,$$

while the point is in T because

$$\bar{x} - (w_i \bar{y}_i + M(1 - \bar{y}_i)) = \left(\frac{1}{k}w_1 + \frac{k-1}{k}M\right) - \left(\frac{1}{k}w_i + \frac{k-1}{k}M\right) = \frac{1}{k}(w_1 - w_i) \leq 0$$

for all $i \in [k]$. Therefore, $(\bar{x}, \bar{y}) \in T_C \setminus S_C$, which completes the proof. $\square$

Chapter 5

Summary and Conclusions

In the thesis, we identify a new connection between tree ensemble optimization and multilinear optimization. Taking a multilinear optimization view on tree ensemble optimization allows us to derive improved formulations for this problem. Reciprocally, studying multilinear polytopes through the lens of tree ensembles allow us to develop new convex hull results for these sets. Computational experiments show that the formulations we develop have distinct computational advantages over existing formulations based on the modeling of piecewise-linear functions and/or existing tree ensemble formulations. In particular, these formulations provide tighter relaxations and improve the solution times through branch-and-bound of most instances.

We provide new and improved formulations for piecewise polyhedral relaxations of multilinear optimization problems. For the case of regular hyper-rectangular partitions, our formulations improve those proposed in the literature. Further, for the case where partitions are not regular, our results yield the first locally ideal formulations for modeling piecewise polyhedral relaxations in the space of both indicator and positioning variables. Such formulations have distinct advantages in branching. Our computational experiments show that these results are directly applicable to solvers such as ALPINE, which is an open source mixed-integer nonlinear programming (MINLP) solver. In particular, when applied to a large collection of hard multilinear and polynomial optimization instances, these results reduce by a factor of approximately 3 the number of instances that ALPINE cannot solve within an hour. They reduce here a bit more by a factor of approximately 4 the number of instances that SCIP, another open-source solver that uses a different solution approach, cannot solve within an hour.

Finally, we study solution methodologies based on discrete optimization for symbolic regression. We provide new valid inequalities for MINLP formulations and develop novel MINLP-based local search heuristics. In computational experiments, we show that valid inequalities improve solution times compared to existing MINLP formulations in finding global solutions when the size of the expression tree is small. Experiments on discovering known physics formulas from data demonstrate that the heuristic we develop outperforms the best known algorithm from the literature.

Throughout this dissertation, we develop convex hull descriptions for a certain class of mixed-integer linear/nonlinear sets and show that the applications of these results improve computations in a variety of optimization problems including tree ensemble optimization problems, multi-commodity transportation problems, and multilinear problems. As a result, a direction of future research is to study computationally effective relaxations of those convex hulls because complete convex hull descriptions require relatively many variables and/or constraints. Another direction is to study tree ensemble optimization problems when tree ensembles appear as constraints of a model. A direction of future research for symbolic regression is to integrate our methods with AI Feynman or other decomposition-based methods because our methods are effective at finding relatively simple symbolic expressions.

References

Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020. doi: 10.1007/s10107-020-01474-5.

Gustavo Angulo, Shabbir Ahmed, Santanu S Dey, and Volker Kaibel. Forbidden vertices. *Mathematics of Operations Research*, 40(2):350–360, 2015. doi: 10.1287/moor.2014.0673.

Vernon Austel, Sanjeeb Dash, Oktay Gunluk, Lior Horesh, Leo Liberti, Giacomo Nannicini, and Baruch Schieber. Globally optimal symbolic regression. In *NIPS 2017 Symposium on Interpretable Machine Learning*, 2017.

Vernon Austel, Cristina Cornelio, Sanjeeb Dash, Joao Goncalves, Lior Horesh, Tyler Josephson, and Nimrod Megiddo. Symbolic regression using mixed-integer nonlinear optimization, 2020.

Egon Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486, 1985. doi: https://doi.org/10.1137/0606047.

Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, 1998. doi: 10.1016/S0166-218X(98)00136-X.

Radu Baltean-Lugojan and Ruth Misener. Piecewise parametric structure in the pooling problem: from sparse strongly-polynomial solutions to NP-hardness. *Journal of Global Optimization*, 71(4):655–690, 2018. doi: 10.1007/s10898-017-0577-y.

Xiaowei Bao, Aida Khajavirad, Nikolaos V Sahinidis, and Mohit Tawarmalani. Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, 7(1):1–37, 2015. doi: 10.1007/s12532-014-0073-z.

Andreas Bärmann, Alexander Martin, and Oskar Schneider. The bipartite boolean quadric polytope with multiple-choice constraints. arXiv preprint, 2020.

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

Dimitris Bertsimas, Allison O'Hair, Stephen Relyea, and John Silberholz. An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531, 2016. doi: 10.1287/mnsc.2015.2363.

Dimitris Bertsimas, Jack Dunn, and Nishanth Mundru. Optimal prescriptive trees. *INFORMS Journal on Optimization*, pages ijoo–2018, 2019.

Daniel Bienstock and Gonzalo Munoz. LP formulations for polynomial optimization problems. *SIAM Journal on Optimization*, 28(2):1121–1150, 2018. doi: 10.1137/15M1054079.

Max Biggs, Rim Hariss, and Georgia Perakis. Optimizing objective functions determined from random forests. Available at SSRN 2986630, 2017.

Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi: 10.1137/S0097539793251219.

Leo Breiman, Jerome Friedman, Charles J Stone, and RA Olshen. *Classification and Regression Trees*. CRC Press, 1984.

Christoph Buchheim, Yves Crama, and Elisabeth Rodríguez-Heck. Berge-acyclic multilinear 0–1 optimization problems. *European Journal of Operational Research*, 273 (1):102–107, 2019. doi: 10.1016/j.ejor.2018.07.045.

Sebastián Ceria and João Soares. Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86(3):595–614, 1999. doi: 10.1007/s101070050106.

Yi-Chun Chen and Velibor V Mišić. Assortment optimization under the decision forest model. Available at SSRN 3812654, 2021.

Yi-Chun Chen and Velibor V Mišić. Decision forest: A nonparametric approach to modeling irrational choice. *Management Science*, 2022. doi: 10.1287/mnsc.2021.4256.

Maxime C Cohen, Ngai-Hang Zachary Leung, Kiran Panchamgam, Georgia Perakis, and Anthony Smith. The impact of linear optimization on promotion planning. *Operations Research*, 65(2):446–468, 2017. doi: 10.1287/opre.2016.1573.

Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming*, volume 271. Springer, 2014.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009. doi: 10.1016/j.dss.2009.05.016.

Alison Cozad. *Data- and theory-driven techniques for surrogate-based optimization*. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, 2014.

Alison Cozad and Nikolaos V Sahinidis. A global MINLP approach to symbolic regression. *Mathematical Programming*, 170(1):97–119, 2018. doi: 10.1007/s10107-018-1289-x.

Yves Crama. Concave extensions for nonlinear 0–1 maximization problems. *Mathematical Programming*, 61(1):53–60, 1993. doi: 10.1007/BF01582138.

Yves Crama and Elisabeth Rodríguez-Heck. A class of valid inequalities for multilinear 0–1 optimization problems. *Discrete Optimization*, 25:28–47, 2017. doi: 10.1016/j.disopt.2017.02.001.

Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases, 2020.

Sanjeeb Dash, Oktay Günlük, and Dennis Wei. Boolean decision rules via column generation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/743394beff4b1282ba735e5e3723ed74-Paper.pdf.

C Deepa, K Sathiya Kumari, and V Pream Sudha. A tree based model for high performance concrete mix design. *International Journal of Engineering Science and Technology*, 2(9):4640–4646, 2010. URL http://www.idc-online.com/technical_references/pdfs/civil_engineering/A%20TREE%20BASED.pdf.

Alberto Del Pia and Aida Khajavirad. On decomposability of multilinear sets. *Mathematical Programming*, 170(2):387–415, 2018a. doi: 10.1007/s10107-017-1158-z.

Alberto Del Pia and Aida Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28(2):1049–1076, 2018b. doi: 10.1137/16M1095998.

Arthur Delarue, Ross Anderson, and Christian Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *Advances in Neural Information Processing Systems*, 33:609–620, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/06a9d51e04213572ef0720dd27a84792-Abstract.html.

Alin Dobra and Johannes Gehrke. Secret: A scalable linear regression tree algorithm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 481–487, 2002. doi: abs/10.1145/775047.775117.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi: 10.1007/s101070100263.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

Thomas Duriez, Steven L Brunton, and Bernd R Noack. *Machine learning control-taming nonlinear dynamics and turbulence.* Springer, 2017. doi: 10.1007/978-3-319-40624-4.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, 2014. URL https://www.jmlr.org/papers/volume15/delgado14a/delgado14a.pdf.

Kris Johnson Ferreira, Bin Hong Alex Lee, and David Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88, 2016. doi: 10.1287/msom.2015.0561.

Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003. doi: 10.1007/s10107-003-0395-5.

Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical Report 18-26, Zuse Institute Berlin, 2018. URL https://nbn-resolving.de/urn:nbn:de:0297-zib-69361.

Mika Goos, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM Journal on Computing*, 47(1):241–269, 2018. doi: 10.1137/16M109884X. URL http://www.siam.org/journals/sicomp/47-1/M109884.html.

Bernd Gross and Peter Roosen. Total process optimization in chemical engineering with evolutionary algorithms. *Computers & Chemical Engineering*, 22:S229–S236, 1998. doi: 10.1016/S0098-1354(98)00059-3.

Akshay Gupte, Thomas Kalinowski, Fabian Rigterink, and Hamish Waterer. Extended formulations for convex hulls of some bilinear functions. *Discrete Optimization*, 36:100569, 2020. doi: 10.1016/j.disopt.2020.100569.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2021. URL http://www.gurobi.com.

Taotao He and Mohit Tawarmalani. A new framework to relax composite functions in nonlinear programs. *Mathematical Programming*, 190(1):427–466, 2021. doi: 10.1007/s10107-020-01541-x.

Taotao He and Mohit Tawarmalani. MIP relaxations of composite functions, 2022.

Manuel Herrera, Luís Torgo, Joaquín Izquierdo, and Rafael Pérez-García. Predictive models for forecasting hourly urban water demand. *Journal of Hydrology*, 387(1-2): 141–150, 2010. doi: 10.1016/j.jhydrol.2010.04.005.

Alan J Hoffman. Total unimodularity and combinatorial theorems. *Linear Algebra and its Applications*, 13(1-2):103–108, 1976. doi: 10.1016/0024-3795(76)90047-1.

Lior Horesh, Leo Liberti, and Haim Avron. Globally optimal minlp formulation for symbolic regression. Technical report, IBM Research, 2016. URL https://dominoweb.draco.res.ibm.com/reports/rc25620.pdf.

Joey Huchette and Juan Pablo Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. *Operations Research* Forthcoming, 2022. URL http://arxiv.org/abs/1708.00050.

Toshimde Ibaraki. Integer programming formulation of combinatorial optimization problems. *Discrete Mathematics*, 16(1):39–52, 1976. doi: 10.1016/0012-365X(76)90091-1.

IBM. IBM ILOG CPLEX optimization studio, 2019. URL www.cplex.com.

Lukas Kammerer, Gabriel Kronberger, Bogdan Burlacu, Stephan M Winkler, Michael Kommenda, and Michael Affenzeller. Symbolic regression by exhaustive search: Reducing the search space using syntactical constraints and efficient semantic structure deduplication. In *Genetic Programming Theory and Practice XVII*, pages 79–99. Springer, 2020. doi: 10.1007/978-3-030-39958-0_5.

Jinhak Kim, Mohit Tawarmalani, and Jean-Philippe P Richard. Convexification of permutation-invariant sets and an application to sparse principal component analysis. *Mathematics of Operations Research*, 2021. doi: 10.1287/moor.2021.1219.

Jongeun Kim, Bijan Taslimi, Jean-Philippe P. Richard, and Mohit Tawarmalani. Polyhedral results for tree ensembles optimization. In *2019 INFORMS Annual Meeting*, 2019.

Tamás Kis and Markó Horváth. Ideal, non-extended formulations for disjunctive constraints admitting a network representation. *Mathematical Programming*, 2021. doi: 10.1007/s10107-021-01652-z.

Gabriel Kronberger, Lukas Kammerer, Bogdan Burlacu, Stephan M Winkler, Michael Kommenda, and Michael Affenzeller. Cluster analysis of a symbolic regression search space. In *Genetic Programming Theory and Practice XVI*, pages 85–102. Springer, 2019. doi: 10.1007/978-3-030-04735-1_5.

T Lehnhäuser and M Schäfer. A numerical approach for shape optimization of fluid flow domains. *Computer methods in applied mechanics and engineering*, 194(50-52): 5221–5241, 2005. doi: 10.1016/S0098-1354(98)00059-3.

James Luedtke, Mahdi Namazifar, and Jeff Linderoth. Some results on the strength of relaxations of multilinear functions. *Mathematical Programming*, 136(2):325–351, 2012. doi: 10.1007/s10107-012-0606-z.

Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274, 2015. doi: 10.1021/ci500747n.

Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10(1): 147–175, 1976. doi: 10.1007/BF01580665.

Arnaud Mignan and Marco Broccardo. Neural network applications in earthquake prediction (1994–2019): Meta-analytic and statistical insights on their limitations. *Seismological Research Letters*, 91(4):2330–2342, 2020. doi: 10.1785/0220200021.

Ruth Misener and Christodoulos A Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. *Mathematical Programming*, 136(1):155–182, 2012. doi: 10.1007/s10107-012-0555-6.

Ruth Misener and Christodoulos A Floudas. Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

Velibor V Mišić. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020. doi: 10.1287/opre.2019.1928.

Miten Mistry, Dimitrios Letsios, Gerhard Krennrich, Robert M Lee, and Ruth Misener. Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded. *INFORMS Journal on Computing*, 33(3):1103–1119, 2021. doi: 10.1287/ijoc.2020.0993.

Harsha Nagarajan, Mowen Lu, Emre Yamangil, and Russell Bent. Tightening mccormick relaxations for nonlinear programs via dynamic multivariate partitioning. In *International conference on principles and practice of constraint programming*, pages 369–387. Springer, 2016. doi: 10.1007/978-3-319-44953-1_24.

Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. *Journal of Global Optimization*, 74(4):639–675, 2019. doi: 10.1007/s10898-018-00734-1.

Pascal Neumann, Liwei Cao, Danilo Russo, Vassilios S Vassiliadis, and Alexei A Lapkin. A new formulation for symbolic regression to identify physico-chemical laws from experimental data. *Chemical Engineering Journal*, page 123412, 2019. doi: 10.1016/j.cej.2019.123412.

Miguel Nicolau and James McDermott. Genetic programming symbolic regression: What is the prior on the prediction? In *Genetic Programming Theory and Practice XVII*, pages 201–225. Springer, 2020. doi: 10.1007/978-3-030-39958-0_11.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997. doi: 10.1016/S0167-7152(96)00140-X.

Panos M Pardalos, Varvara Rasskazova, Michael N Vrahatis, et al. *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems.* Springer, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011. URL https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https://githubhelp.com.

Nikolaos Ploskas and Nikolaos V Sahinidis. Review and comparison of algorithms and software for mixed-integer derivative-free optimization. *Journal of Global Optimization*, pages 1–30, 2021. doi: 10.1007/s10898-021-01085-0.

Xinbo Qi, Guofeng Chen, Yong Li, Xuan Cheng, and Changpeng Li. Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives. *Engineering*, 5(4):721–729, 2019. doi: 10.1016/j.eng.2019.04.012.

John R Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific, 1992. doi: 10.1142/9789814536271.

Anatoliy D Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1997. doi: 10.1023/A:1008217604285.

Ralph Tyrell Rockafellar. *Convex analysis.* Princeton university press, 1970.

N. V. Sahinidis. *BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017. URL https://sahinidis.coe.gatech.edu/baron.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. ISSN 0036-8075. doi: 10.1126/science.1165893.

Alexander Schrijver. Short proofs on the matching polyhedron. *Journal of Combinatorial Theory, Series B*, 34(1):104–108, 1983. doi: 10.1016/0095-8956(83)90011-4.

Hanif D Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta mathematica vietnamica*, 22(1):245–270, 1997. URL http://journals.math.ac.vn/acta/pdf/9701245.pdf.

Robert A Stubbs and Sanjay Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical programming*, 86(3):515–532, 1999. doi: 10.1007/s101070050103.

Guolong Su, Dennis Wei, Kush R. Varshney, and Dmitry M. Malioutov. Learning sparse two-level boolean rules. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016. doi: 10.1109/MLSP.2016.7738856.

Kaarthik Sundar, Harsha Nagarajan, Jeff Linderoth, Site Wang, and Russell Bent. Piecewise polyhedral formulations for a multilinear term. *Operations Research Letters*, 49(1):144–149, 2021a. doi: 10.1016/j.orl.2020.12.002.

Kaarthik Sundar, Sujeevraja Sanjeevi, and Harsha Nagarajan. Sequence of polyhedral relaxations for nonlinear univariate functions. *Optimization and Engineering*, pages 1–18, 2021b. doi: 10.1007/s11081-021-09609-z.

Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003. doi: 10.1021/ci034160g.

Bashar Tarawneh, Wassel AL Bodour, and Khaled Al Ajmi. Intelligent computing based formulas to predict the settlement of shallow foundations on cohesionless soils. *The Open Civil Engineering Journal*, 13(1), 2019. doi: 10.2174/1874149501913010001.

Mohit Tawarmalani. Inclusion certificates and simultaneous convexification of functions. Available at Optimization Online, 2010. URL http://www.optimization-online.org/DB_HTML/2010/09/2722.html.

Mohit Tawarmalani and Nikolaos V Sahinidis. Convex extensions and envelopes of lower semi-continuous functions. *Mathematical Programming*, 93(2):247–263, 2002. doi: 10.1007/s10107-002-0308-z.

Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. Entmoot: a framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021. doi: 10.1016/j.compchemeng.2021.107343.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity, 2020.

Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *Siam Review*, 57(1):3–57, 2015. doi: 10.1137/130915303.

Juan Pablo Vielma and George L Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1):49–72, 2011. doi: 10.1007/s10107-009-0295-4.

Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations research*, 58(2):303–315, 2010. doi: 10.1287/opre.1090.0721.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. doi: 10.1007/s10107-004-0559-y.

Yiqun Wang, Nicholas Wagner, and James M Rondinelli. Symbolic regression in materials science. *MRS Communications*, 9(3):793–805, 2019. doi: 10.1557/mrc.2019.85.

David R. White, James Mcdermott, Mauro Castelli, Luca Manzoni, Brian W. Goldman, Gabriel Kronberger, Wojciech Jaundefinedkowski, Una-May O'Reilly, and Sean Luke. Better GP benchmarks: Community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3—-29, March 2013. ISSN 1389-2576. doi: 10.1007/s10710-012-9177-2.

Yibo Xu, Warren Adams, and Akshay Gupte. Polyhedral analysis of symmetric multilinear polynomials over box constraints. *arXiv preprint arXiv:2012.06394*, 2021. doi: 10.48550/arXiv.2012.06394.

I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998. doi: 10.1016/S0008-8846(98)00165-3.