A Reformulation Technique to Solve Polynomial Optimization Problems with Separable Objective Functions of Bounded Integer Variables

Pitchaya Wiratchotisatian[1] and Andrew C. Trapp[2]

[1]Department of Statistics, Faculty of Science, Khon Kaen University
[2]WPI Business School, Worcester Polytechnic Institute

Real-world problems are often nonconvex and involve integer variables, representing vexing challenges to be tackled using state-of-the-art solvers. We introduce a mathematical identity-based reformulation of a class of polynomial integer nonlinear optimization (PINLO) problems using a technique that linearizes polynomial functions of separable and bounded integer variables of any degree. We also introduce an alternative reformulation and conduct computational experiments to understand their performance against leading commercial global optimization solvers. Computational experiments reveal that our integer linear optimization (ILO) reformulations are computationally tractable for solving large PINLO problems via Gurobi (up to 10,000 constraints and 20,000 variables). This is much larger than current leading commercial global optimization solvers such as BARON, thereby demonstrating its promise for use in real-world applications of integer linear optimization with a polynomial objective function.

Keywords: *Polynomial Integer Optimization, Reformulation, Linearization, Integer Optimization*

---

1 Introduction

Integer optimization (IO) has seen widespread use in solving challenging decision problems due to its expressivity and ability to characterize constrained decisions under an objective function to be optimized. Advances in algorithmic development and computing over the past several decades have seen profound increases in the potential of commercial optimization solvers [13]. Even so, many real world decision problems with nonconvex functions represent vexing challenges to solve to global optimality, even with state-of-the-art global optimization solvers.

We study a broad class of polynomial integer nonlinear optimization (PINLO) problems with continuous variables in linear functional components:

$$\begin{array}{lll} \text{maximize} & \sum_{k=1}^{p}\sum_{j=1}^{n_x} c_{kj}x_j^k + \sum_{\ell=1}^{n_y} c_\ell y_\ell & \\ \text{such that} & \sum_{j=1}^{n_x} a_{ij}x_j + \sum_{\ell=1}^{n_y} a_{i\ell}y_\ell \le b_i, & i = 1,\ldots,m, \\ & x_j \in \mathbf{X}_j = \{x_j^L \le x_j \le x_j^U,\ x_j \in \mathbb{Z}_{\ge 0}\}, & j = 1,\ldots,n_x, \\ & y_\ell \in \mathbf{Y}_\ell = \{y_\ell^L \le y_\ell \le y_\ell^U,\ y_\ell \in \mathbb{R}\}, & \ell = 1,\ldots,n_y, \end{array} \qquad (\texttt{PINLO})$$

where the objective function is a separable polynomial function, constraints are linear $\forall\ i = 1,\ldots,m$, x_j^L and x_j^U are lower and upper bounds of nonnegative integer $x_j\ \forall\ j = 1,\ldots,n_x$, and x_j^k is the k^{th} degree polynomial of $x_j\ \forall\ j = 1,\ldots,n_x,\ \forall\ k = 1,\ldots,p$. When $n_y > 0$, continuous variables appear in linear expressions with lower and upper bounds of y_ℓ^L and $y_\ell^U\ \forall\ \ell = 1,\ldots,n_y$, respectively. We will refer to this specific class of (PINLO) formulation as PINLO throughout this study.

Many real-world applications of nonlinear integer optimization feature a quadratic objective function subject to a set of linear constraints, often called mixed-integer quadratic programming (MIQP), and when there are no continuous variables, (pure) integer quadratic programming (IQP) [1, 41]. These problems can be viewed as a generalization of (mixed-)integer linear optimization with a nonlinear objective function [24]. Thus, (PINLO) formulation encompasses all integer linear optimization (ILO) problems, including applications in scheduling, planning and network flows. PINLO appears (often in quadratic form) in a variety of applications, such as quadratic knapsack problems [14–16, 46], two-stage quadratic stochastic programs [51], multicommodity network flow problems [56], portfolio selection [25, 30, 31, 64], heat transfer processes [52], and goal programming with quadratic deviation penalties [47, 57].

Many exact methods for solving (PINLO) formulation have been proposed. Branching-based methods include branch-and-bound (BB) [38, 39, 50, 58], branch-and-reduce [11, 53, 54, 59, 60], and αBB [3–6]. Methods that reformulate PINLO problems into ILO problems include the use of sets of binary variables with a certain special structure, called special ordered sets of type 2 (SOS2) [9, 10], as well as piecewise linear functions [8, 27, 34, 35, 40, 63].

Other exact algorithms include the use of concavity cutting planes [12, 17, 61], a hybrid method that combines dynamic programming and branch-and-bound approaches to produce an algorithm for solving separable discrete optimization [49], an algorithm to simplify nonseparable functions [36], a Lagrangian decomposition technique [33], and expressing nonconvexity in the objective and constraint functions as the sum of nonconvex univariate functions [21]. Moreover, [18, 22–24, 32] review multiple global optimization approaches for the general nonconvex optimization problem.

This paper presents two reformulations of a class of PINLO problems with separable and bounded integers and any degree of polynomial to ILO problems. We demonstrate how to exploit an algebraic identity presented in Sect. 2.1 in our ILO reformulation that expresses values of polynomial integers as the summation of cumulative weights. We also introduce an alternative reformulation that uses precomputed weights to express polynomial values. We demonstrate the equivalence of (PINLO) formulation and the proposed ILO reformulations. We subsequently conduct comparative computational experiments of PINLO and ILO reformulations. The computational experiments reveal that the cumulative-weight ILO outperforms precomputed-weight ILO in larger problem dimensions. Further, the cumulative-weight ILO reformulation outperforms PINLO using the state-of-the-art commercial solvers BARON [55, 60] and Gurobi [29].

The remainder of this paper is structured as follows. Sect. 2 presents theoretical concepts to linearize the specific class of (PINLO) formulation. Sect. 3 presents computational experiments to compare our approaches via leading commercial optimization solvers. Sect. 4 concludes our study.

2 Linearization of PINLO

We introduce a linearization of (PINLO) formulation using recurrence relations and the series identity of positive integer powers of degree p. We first present the identity for our linearization in Sect. 2.1 which can be derived from Faulhaber's formula [37] as well as the binomial theorem. We next outline the steps for PINLO-to-ILO linearization which expresses a polynomial of integer variables as the cumulative summation of the derived formulations in Sect. 2.2. Lastly, for the comparative results, we present another linearization of PINLO-to-ILO which expresses polynomial functions of integer variables as the summation of products of precomputed weights and binary variables in Sect. 2.3.

2.1 Derivation of The Finite Summation Identity

To find an expression of the p^{th} power of any positive integer n, we explore the reformulation of polynomial terms using the finite summation identity that equals n^p. One way to formulate n^p as a finite summation of n summation terms is via the recursion of the finite summation of the p^{th} power of the first n positive integers. The finite summation identity of n^2 is equal to the sum of the first n positive odd integers,

$$n^2 = \sum_{i=1}^{n}(2i-1),$$

which can be derived from this well-known identity,

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

In general, we can use *Faulhaber's formula* [37] to express the sum of the p^{th} power of the first n positive integers in terms of the Bernoulli numbers, as follows:

$$\sum_{i=1}^{n} i^p = \frac{1}{p+1}\sum_{k=0}^{p}(-1)^k \binom{p+1}{k} B_k n^{p-k+1}, \tag{1}$$

where B_k is the Bernoulli number[1] with $B_1 = -\frac{1}{2}$.

Another way, perhaps simpler, is to apply the binomial theorem to derive the finite summation of n^p. For any integer i, the binomial theorem states that

$$(i+1)^{p+1} = \sum_{k=0}^{p+1} \binom{p+1}{k} i^k. \tag{2}$$

By moving the last summand to the left hand side, this yields the following expression:

$$(i+1)^{p+1} - i^{p+1} = \sum_{k=0}^{p} \binom{p+1}{k} i^k. \tag{3}$$

Substituting $i = -1, -2, \ldots, -n$ to (3) and summing, the left-hand side telescopes to $-(-n)^{p+1}$ and we obtain

$$-(-n)^{p+1} = \sum_{k=0}^{p}(-1)^k \binom{p+1}{k} \left[1^k + \cdots + n^k\right]. \tag{4}$$

Multiplying both sides of (4) by $(-1)^p$, we obtain the following identity

$$n^{p+1} = \sum_{k=0}^{p}(-1)^{p+k} \binom{p+1}{k} \sum_{i=1}^{n} i^k, \tag{5}$$

which can be used to express the p^{th} power of any positive integer n in term of the summation of n terms as

$$n^p = \sum_{i=1}^{n}\sum_{k=0}^{p-1}(-1)^{p+k-1} \binom{p}{k} i^k. \tag{6}$$

[1] Bernoulli numbers [65] are a sequence of rational numbers of importance in number theory, with the first few Bernoulli numbers being $B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_4 = -\frac{1}{30}, \ldots$

For example, $n^2 = \sum_{i=1}^{n}(-1)i^0 + \sum_{i=1}^{n}(2)i^1 = \sum_{i=1}^{n}(2i-1)$. We next apply (6) to linearize any positive integer power of degree p.

2.2 PINLO-to-ILO Linearization via Cumulative Summation to Express Powers of Integer Variables

We introduce a linearization of polynomial nonlinear terms of bounded integers. We outline the reformulation of the PINLO-to-ILO reformulation in three steps. The first step is to define binary variables that collectively represent all integer variable values, and through constraints enforce the integer variable values by summing over binary variables. The second step is to ensure that, depending on the value of the integer variable, the appropriate binary variables are activated. The last step is the derivation of appropriate weights that, when aggregated, are an equivalent representation of the variable values in the respective polynomial expression.

First step of reformulation. We express the value of each nonnegative integer $x_j \ \forall \ j = 1, \ldots, n_x$ in (`PINLO`) formulation as the summation of binary variables $x_{j,d}$ where the index $d \in [1, x_j^U] \subset \mathbb{Z}_{\geq 0}$ represents the occurrence of the binary variables for x_j. We define binary variables $x_{j,d}$ taking a value of 1 if $x_j \geq d$ for integer values $d \geq 1$, respectively, and 0 otherwise. This means that if $x_j = k$ for some positive integer k, then all k binary variables $x_{j,1}, \ldots, x_{j,k}$ are activated to 1 because $x_j \geq k \geq \cdots \geq 1$. Fig. 1 demonstrates the activation of $x_{j,d}$ when $x_j = 8$. The activation is applied through constraint sets (8) and (9) that we next define.

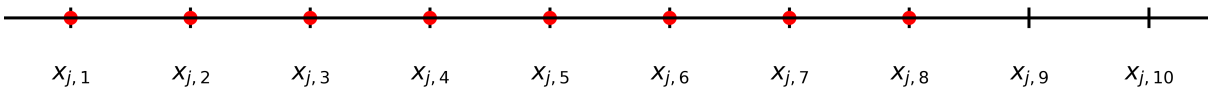


Figure 1: Illustrating activated $x_{j,d}$ in red when $x_j = 8$.

Second step of reformulation. The integer decision variables x_j are now represented as the sum of $x_{j,d}$, where

$$x_j = \sum_{d=1}^{x_j^U} x_{j,d}. \tag{7}$$

We use disjunctive constraints to ensure that the binary variables take appropriate values to represent the integer variable values x_j. That is, through the following two sets of inequalities, $x_{j,d} = 1$ if and only if $x_j \geq d$ and 0 otherwise, for integer value $1 \leq d \leq x_j^U$:

$$x_j - d \leq M_{j,d} x_{j,d} - 1 \quad \forall \ j = 1, \ldots, n_x, \ \forall \ d = 1, \ldots, x_j^U, \tag{8}$$

$$x_j - d \geq m_{j,d}(1 - x_{j,d}) \quad \forall \ j = 1, \ldots, n_x, \ \forall \ d = 1, \ldots, x_j^U. \tag{9}$$

The value of each $M_{j,d}$ is the upper bound of x_j and the value of each $m_{j,d}$ is the lower bound of x_j minus the upper bound of x_j. If $x_j \geq d$, $x_{j,d}$ must be one, otherwise constraint set (8) is violated. If $x_j < d$, $x_{j,d}$ must be zero, otherwise constraint set (9) is violated.

Final step of reformulation. To enforce the polynomial value, we introduce special weights via the identity (6) that states the equivalence of the p^{th} power of a positive integer number n^p and the sum of n positive integer numbers. Denote $w_{j,d}$ to be a weight value of $x_{j,d}$ $\forall\ d \in \{1, \ldots, x_j^U\}$ taking a value of

$$w_{j,d} = \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} d^k \quad \forall\ d \in \{1, \ldots, x_j^U\}, \tag{10}$$

so that their cumulative sum of the product of $w_{j,d}$ and $x_{j,d}$ over $d = 1, \ldots, x_j^U$ equals to x_j^p. Equivalently, we can view the weight $w_{j,d}$ as an increment of an additional unit increase of x_j^p.

Through these three steps, our reformulation (`ILO1`) transfers the polynomial expression x_j^p to an equivalent linear expression:

$$x_j^p = \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d} \quad \forall\ j = 1, \ldots, n_x. \tag{ILO1}$$

Our (`ILO1`) reformulation offers a new way to represent nonlinear expressions, in particular polynomial expressions, in a linear manner which can be implemented directly using any state-of-the-art ILO solver. We demonstrate the equivalence of the PINLO and reformulated ILO objective functions in Proposition 1.

Proposition 1 *For $x_j \in \mathbf{X}_j$, $x_j^p = \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d}$.*

Proof. Without loss of generality, assume $x_j = v$ for an arbitrary $v \in \{1, \ldots, x_j^U\}$ in the domain of $x_j \in \mathbf{X}_j$:

$$\begin{aligned}
\sum_{d=1}^{x_j^U} w_{j,d} x_{j,d} &= \sum_{d=1}^{x_j^U} \left(\sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} d^k \right) x_{j,d} \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[1^k \cdot x_{j,1} + \cdots + (x_j^U)^k \cdot x_{j,x_j^U} \right] \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[x_{j,1} + \cdots + v^k \cdot x_{j,v} \right], \ \{\text{by (8) and (9)}\} \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[1 + \cdots + v^k \right] \\
&= v^p, \ \{\text{by (6)}\}
\end{aligned}$$

as the choice of v was arbitrary, this completes the proof. ■

Proposition 1 implies the equivalence of the objective functions of (`PINLO`) formulation and (`ILO1`) formulation, that is,

$$\sum_{j=1}^{n_x} c_{kj} x_j^p = \sum_{j=1}^{n_x} c_{kj} \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d}.$$

2.3 PINLO-to-ILO Linearization via Precomputed Weights to Express Powers of Integer Variables

We compare (`ILO1`) with an alternative reformulation (`ILO2`) that linearizes polynomial terms using precomputed weights for the value of polynomial integers. The (`ILO2`) formulation is a conventional method to convert integer polynomial optimization problems into 0–1 integer linear optimization problems [7]. We express the p^{th} power of x_j in (`PINLO`) formulation in terms of the summation of precomputed weights

multiplied by binary variable $x_{j,q}\ \forall\ q = 0, \ldots, x_j^U$ where $x_{j,q} = 1$ if $x_j = q$, and 0 otherwise. As a result, we can express x_j^p as

$$x_j^p = \sum_{q=0}^{x_j^U} q^p x_{j,q}, \quad \forall\ j = 1, \ldots, n_x. \tag{ILO2}$$

To enforce (ILO2), we first ensure that exactly one binary variable is activated by adding the following constraint

$$\sum_{q=0}^{x_j^U} x_{j,q} = 1 \quad \forall\ j = 1, \ldots, n_x. \tag{11}$$

Then, we impose the relationship between x_j and the binary variables in which $x_j = q\ \forall\ j = 1, \ldots, n_x$ if $x_{j,q} = 1$ by adding the following constraint

$$x_j = \sum_{q=0}^{x_j^U} q x_{j,q} \quad \forall\ j = 1, \ldots, n_x. \tag{12}$$

This reformulation is more succinct and potentially more straightforward than the (ILO1) reformulation. We next present computational experiments that compare the performance of our (ILO1) reformulation with (PINLO) formulation as well as the (ILO2) reformulation and its variants in Sect. 3.2.

3 Computational Experiments

We evaluate the performance of our (ILO1) reformulation by comparing with (PINLO) formulation and (ILO2) reformulation using precomputed weights. All experiments were run on NEOS server [19, 20, 28] using BARON 21.1.13 [55, 60] to solve PINLO and ILO. We also solve ILO via Gurobi Optimization 9.1 [29] and Python API with up to 64 GB memory, under Red Hat Enterprise Linux Server 7.3 with kernel version 3.10.0-514.x86_64. Each instance (run) was run with time limit of 3 hours, MIP optimality gap tolerance of 0, with absolute MIP optimality gap of 0, and thread count of 1.

3.1 Data Description

We perform a full factorial design to better understand model performance across solvers, by varying the parameters listed in Table 1. The domain of the integer variables is $[0, x^U]$, where the upper bound $x^U \in \{10, 100\}$. The degree of polynomial objective functions of integer variables in all instances is chosen to be a reasonable degree in polynomial problems $p \in \{2, 3, 5\}$, though it is not prohibitive to increase p to much larger values.

We select the values of model parameters in a similar manner as described in [26]. We set the density of constraints to be 50%. Coefficients in the left-hand side of the constraints are drawn randomly from a discrete uniform distribution $[1, 30]$ and constants in the right-hand side of the constraints are drawn randomly from a discrete uniform distribution $[30, 30 + \sum_{j=1}^{n_x} a_{ij}]$ for $i = 1, \ldots, m$. We set the number of constraints $m \in \{25, 50, 75, 100, 150, 250, 500, 750, 1{,}000\}$. The total number of variables n is controlled by the scale parameter for the problem size $\alpha \in \{0.5, 2\}$ in which $n = \alpha m$.

We generate two classes of instances controlled by the ratio of the number of integer variables to the total number of variables. Instances are pure integer when all variables are integers $\left(\frac{n_x}{n} = 1\right)$ and are mixed integer when the number of integer variables (n_x) is half of the total number of variables

Table 1: Parameters used for generating problem instances.

Parameter	Symbol	Levels
Integer variable upper bound	x^U	$10, 100$
Polynomial degree	p	$2, 3, 5$
Ratio of integer-to-total number of variables	$\frac{n_x}{n}$	$0.5, 1$
Density of objective coefficients	Δ	$0.5, 1$
Number of constraints	m	$25, 50, 75, 100, 150,$ $250, 500, 750, 1{,}000$
Scale parameter for problem size	α	$0.5, 2$

$\left(\frac{n_x}{n} = 0.5\right)$. More specifically, variables in mixed-integer instances consist of 50% integer variables, 25% continuous variables with values between $[0, 10]$, and 25% binary variables. We summarize the model size of the three approaches in Table 2. When (`PINLO`) formulations contain n variables and m constraints, the proposed (`ILO1`) formulations contain at most $n(x^U + 1)$ variables and $m + n(2x^U + 1)$ constraints, while the (`ILO2`) formulations contain at most $n(x^U + 2)$ variables and $m + 2n$ constraints.

Table 2: Comparison of model size generated by the three formulations PINLO, ILO1, and ILO2.

Model Size	PINLO	ILO1	ILO2
Continuous variables (y_ℓ)	n_y	n_y	n_y
Binary variables (x_j)	n_b	n_b	n_b
Integer variables (x_j)	n_x	n_x	n_x
Auxiliary 0-1 variables ($x_{j,d}$)	-	$n_x x^U$	$n_x(x^U + 1)$
Linear inequality constraints	m	m	m
Total sum constraints	-	n_x	$2n_x$
Disjunctive constraints	-	$2n_x x^U$	-

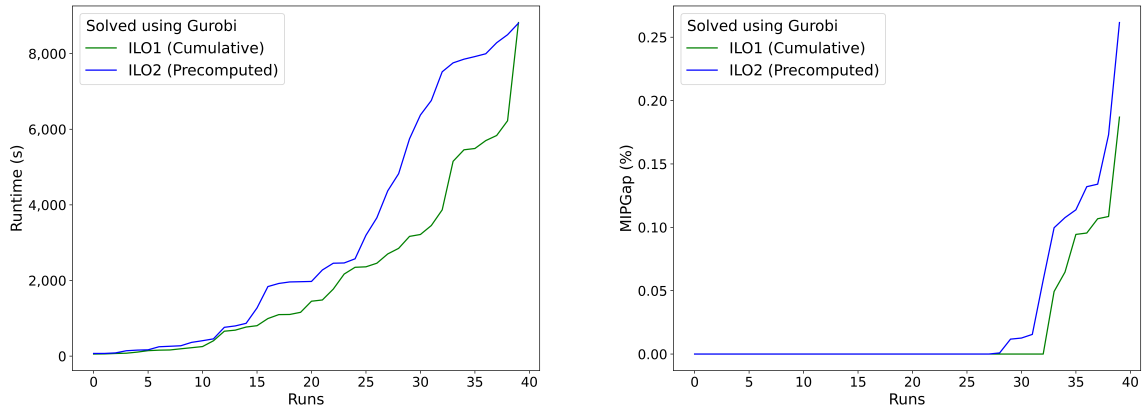
The objective function of every instance contains integer variables, each with degree up to p, as well as continuous and binary variables if the instance is a mixed-integer problem. For example, an objective function of pure integer instances contains up to $n \times p$ integer terms as each integer variable may have degree up to p, whereas an objective function of mixed-integer instances contains $n \times p$ integer terms and $\frac{1}{2}n$ continuous and binary terms. The density of nonzero elements in the objective function is controlled by a Bernoulli probability $\Delta \in \{0.5, 1\}$, where the values of the coefficients are drawn from a discrete uniform distribution $[0, 100]$. The combination results in 432 runs in total, for which we create three replicates and average over each run. We report the computational results in Sect. 3.2.

3.2 Computational Results and Discussion

We conduct four experiments to evaluate the performance of our reformulations. We first compare (`ILO1`) versus (`ILO2`) via the Gurobi solver on large instances. As (`ILO1`) outperforms (`ILO2`) on larger instances, we next compare (`PINLO`) solved via BARON versus (`ILO1`) solved via both BARON and Gurobi. We then increase the size of the instances and evaluate (`ILO1`) via Gurobi on pure and mixed-integer problems, to provide further insights into the computational performance of the (`ILO1`) reformulation. As (`ILO2`) is an established method, we finally compare the performance of (`ILO1`) versus (`ILO2`) as well as two recent extensions [43] via Gurobi, in particular reducing the number of binary variables logarithmically [2, 42, 44, 45, 62], as well as adding incremental formulations to better balance the branch-and-bound tree [66].

We first compare the computational performance of (`ILO1`) and (`ILO2`). We fix every parameter at

their extreme values and vary the upper bound of integer variables $x^U \in \{10, 100\}$, the problem types $\frac{n_x}{n} \in \{0.5, 1\}$, and the number of constraints $m \in \{1{,}000, 2{,}000, \ldots, 10{,}000\}$, resulting in 40 instances. The results of each instance is averaged over three replicates. Fig. 2 plots the results from smaller to larger problem size. After a brief period where the computational performance of the two approaches appear similar, it becomes clear that (`ILO1`) outperforms (`ILO2`) for larger problem instances. Moreover, Fig. 2b shows that (`ILO1`) results in more optimal instances than (`ILO2`).



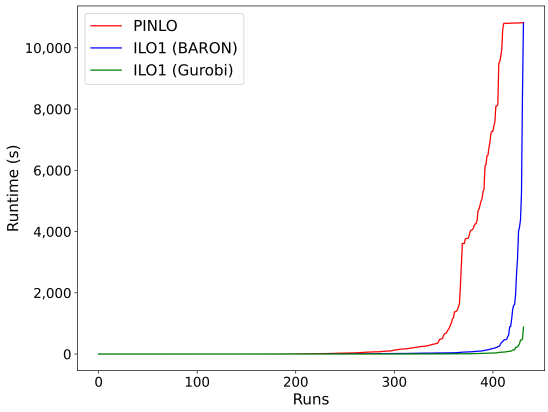
(a) Average runtimes.

(b) Average MIP gaps.

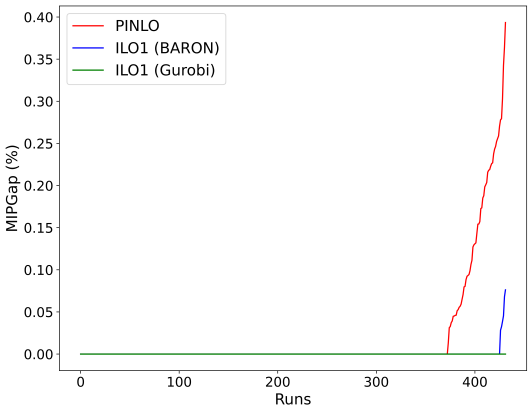
Figure 2: Comparing the computational performance of (`ILO1`) and (`ILO2`) solved via Gurobi on problem instances with up to 10,000 constraints and 20,000 variables.

As (`ILO1`) shows superior computational performance over (`ILO2`) on difficult instances, we now compare the performance of solving (`PINLO`) formulation and (`ILO1`) formulation via BARON, as well as (`ILO1`) formulation via Gurobi. First, we level the playing field of (`PINLO`) formulation and (`ILO1`) formulation by comparing their results from BARON. Fig. 3 shows that (`ILO1`) outperforms (`PINLO`) with respect to the average runtimes and MIP gaps over each run of (`ILO1`) (blue lines) as it results in more runs with lower runtime and MIP gap than (`PINLO`) (red lines). When we solve ILO via Gurobi (green lines), all runs find optimal solutions with 0% MIP gaps with a mean value of 14.6 seconds and a maximum (mean) runtime of less than 900 seconds for all runs. Fig. 3c reveals that when comparing (`ILO1`) solved via either BARON or Gurobi versus (`PINLO`) solved via BARON, that over 100 runs result in notably faster computational performance (positive runtime difference) when comparing run-by-run differences in runtime. Specifically, by considering only instances with average runtime difference of (`ILO1`) and (`PINLO`) strictly greater or less than 1% of the 3-hour time limit (or 108 seconds), (`ILO1`) solved via BARON outperforms PINLO in 130 runs, while (`ILO1`) solved via Gurobi outperforms (`PINLO`) in 134 runs, and for no runs did (`PINLO`) outperform either (`ILO1`).

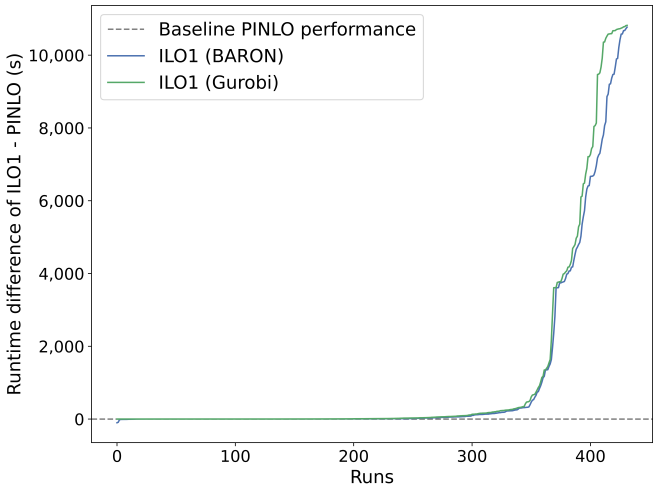
Our experiments show that the computational performance of solving (`ILO1`) via both BARON and Gurobi is superior to solving (`PINLO`) via BARON and (`ILO2`) via Gurobi on hard instances. We now view a broader picture of (`ILO1`) performance. We fix every parameter at their extreme values and increase m up to 10,000, resulting in a maximum model size of 20,000 variables with $\alpha = 2$. We first study the computational performance across different densities of objective function coefficients by varying $\Delta \in \{0.05, 0.25, 0.5, 0.75, 1\}$. Fig. 4a shows that (`ILO1`) finds optimal solutions to all instances of both mixed-integer (red line) and pure integer (blue line) types, with a maximum average runtime of 4,000 and 6,000 seconds, respectively, showing that (`ILO1`) reformulation is relatively insensitive to changes in objective function coefficient densities. We also vary the number of constraints m in increments of 250, up to the maximum size of 10,000 constraints, resulting in $m \in \{25, 50, 75, 100, 150, 250, 500, \ldots, 9{,}750, 10{,}000\}$.



(a) Average runtimes.



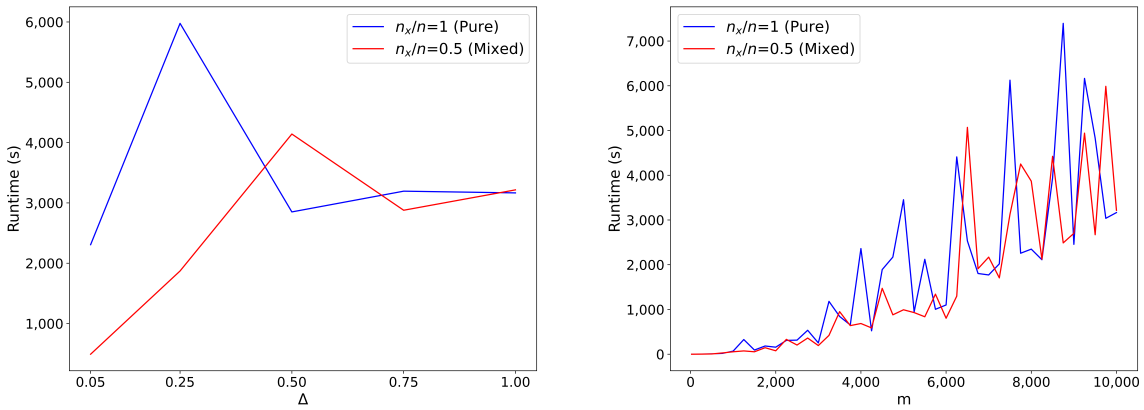
(b) Average MIP gaps.



(c) Run-by-run comparison of the difference in runtime performance (in seconds) of (`ILO1`) over PINLO when solving (`ILO1`) via BARON and Gurobi, where positive runtime difference indicates faster performance of (`ILO1`), the run-by-run performance difference curves are plotted after sorting in ascending order, and the grey curve depicts the baseline performance of (`PINLO`).

Figure 3: Comparative results of 432 runs each averaging over three replicates for PINLO solved via BARON (red line) and (`ILO1`) solved via BARON (blue line) and via Gurobi (green line).

Fig. 4b shows that (ILO1) finds optimal solutions to all instances of both mixed-integer (red line) and pure integer (blue line) types, with a maximum average runtime of 6,000 and 7,400 seconds, respectively, showing that (ILO1) is computationally tractable for solving large problems of size 10,000 constraints and 20,000 variables.



(a) Varying Δ.

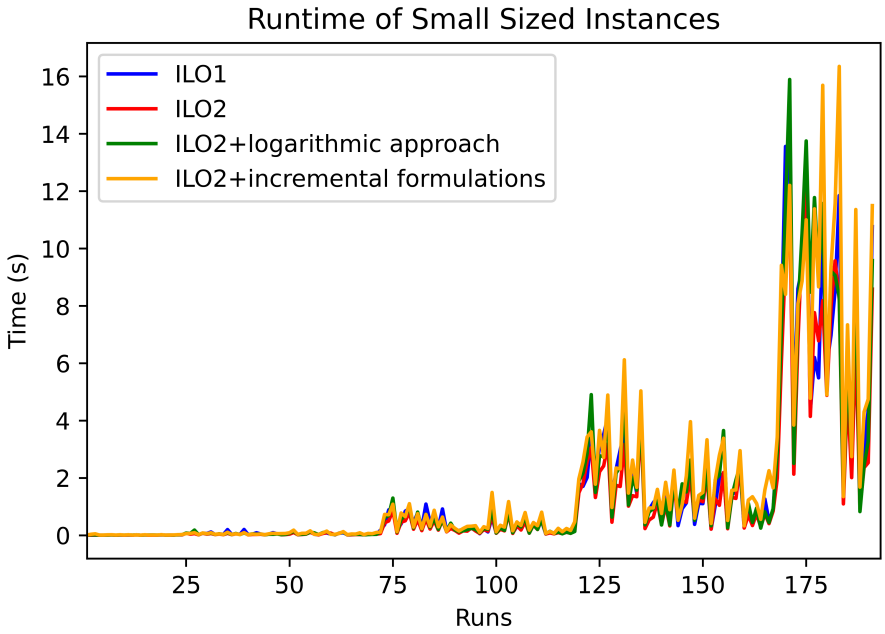
(b) Varying m.

Figure 4: Computational performance of (ILO1) solved via Gurobi on problem instances with up to 10,000 constraints and 20,000 variables where red and blue curves represent mixed-integer and pure integer instances, respectively.

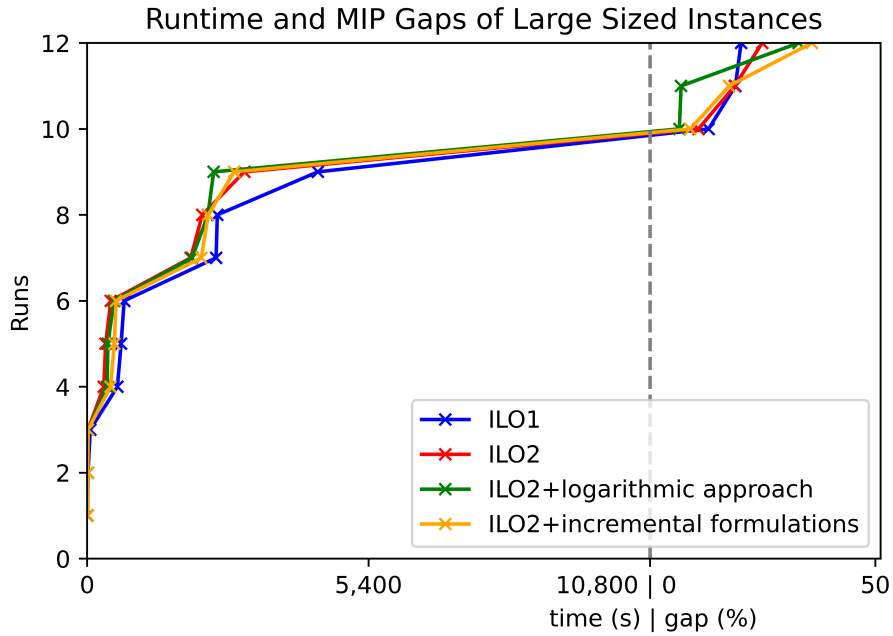
In [43] the two improvements of (ILO2) are shown to generally outperform the conventional method, thus we further compare the computational performance of (ILO1) and (ILO2) and its two variants including reducing the number of binary variables logarithmically, as well as adding incremental formulations to balance the branch-and-bound trees.

We conduct an experiment on small sized instances by varying parameters similar to Table 1, with $m \in \{25, 100, 250, 500\}$. There are 192 combinations in total, each is replicated and run three times. We average the runtime to compare the performance across the four methods. All instances are solved to optimality. For each method, every run is ranked from smallest to largest problem size and the runtimes plotted in Fig. 5a. Shown in Fig. 5a, the runtime of all methods are comparatively similar, and none of the four methods appears to dominate another.

Finally, we conduct another experiment on large sized instances by setting x^U to 100, $\frac{n_x}{n}$ to 0.5, Δ to 0.25, and α to 2, then varying $p \in \{2, 3, 5\}$ and $m \in \{1{,}000, 5{,}000, 10{,}000, 15{,}000\}$. There are 12 combinations of larger instances in total, each is replicated and run three times. We limit the runtime to 3 hours. As a result, all instances are solved to optimality within the time limit, except the largest instances with 15,000 constraints, where none of the four methods is able to solve to optimality. We report average runtimes of optimal instances and average MIP gaps of suboptimal instances in the cumulative distribution plot in Fig. 5b. The results of average runtimes and MIP gaps of each method are ranked in ascending order. While the (ILO2) formulation with logarithmic approach (green line) appears to performs well for some instances (e.g., runs 9 and 11), no method appears to dominate another overall, as every method outperforms others for at least one instance. On larger sized instances (runs 7 through 12), outside of run 9 the (ILO1) formulation performs reasonably well and even outperforms the others in terms of MIP gap for run 12, one of the largest instances with $m = 15{,}000$ and $p = 5$. These results demonstrate that the (ILO1) formulation is both competitive and can sometimes outperform existing methods in terms of competitive MIP gaps for the largest of instances.



(a) $m \in \{25, 100, 250, 500\}$.



(b) $m \in \{1{,}000, 5{,}000, 10{,}000, 15{,}000\}$.

Figure 5: Comparison of (`ILO1`) and (`ILO2`) and its variants. (Top) Runtime performance of each method on small sized instances. (Bottom) Cumulative distribution plot of each method on 12 larger instances.

4 Concluding Remarks

We study polynomial integer nonlinear optimization (PINLO) problems featuring separable polynomial integer objective functions and linear constraints, covering quadratic integer optimization problems and related problem classes with separable polynomial expressions in the objective function. We theoretically derive the finite summation identity motivated from [48] that is used to reformulate PINLO problems to ILO problems via cumulative weighting. Our novel linearization advances linearization techniques for a large class of separable integer nonlinear optimization problems, converting them to ILO problems that we demonstrate are more computationally tractable.

Our computational experiments on synthetically generated test instances show that ILO outperforms PINLO when solved via BARON on the NEOS server. (`ILO1`) is even faster when solved via Gurobi. Moreover, our reformulation is computationally tractable for solving large PINLO problems as we demonstrate the use of (`ILO1`) to solve problems of size 10,000 constraints and 20,000 variables. Our comparison of (`ILO1`) and (`ILO2`) and its enhanced variants [43] shows that (`ILO1`) is competitive with state-of-the-art methods, and in particular at the largest levels of problem size tested.

We believe that the reformulation of a class of PINLO in this study will benefit other applications. Future studies may investigate the performance of (`ILO1`) on instances with larger limits on runtime, number of constraints, and polynomial degree. Also worthwhile to explore, is how to reformulate other nonlinear functions in a similar manner exploiting additional algebraic identities, as well as applying our reformulation to polynomial constraints of the same class.

Acknowledgement

We thank Alfred Scott who inspired initial discussions on reformulating PINLO, and Ryan Killea for his time and assistance with running experiments.

References

[1] MINLPLib (2021) Mixed-integer nonlinear programming library (2021). URL `http://www.minlplib.org/`. Last accessed on February 1, 2023

[2] Adams, W.P., Henry, S.M.: Base-2 expansions for linearizing products of functions of discrete variables. Operations research **60**(6), 1477–1490 (2012)

[3] Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—II. implementation and computational results. Computers & chemical engineering **22**(9), 1159–1179 (1998)

[4] Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: Global optimization of mixed-integer nonlinear problems. AIChE Journal **46**(9), 1769–1797 (2000)

[5] Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. theoretical advances. Computers & Chemical Engineering **22**(9), 1137–1158 (1998)

[6] Androulakis, I.P., Maranas, C.D., Floudas, C.A.: αBB: A global optimization method for general constrained nonconvex problems. Journal of Global Optimization **7**(4), 337–363 (1995)

[7] Balas, E.: Extension de l'algorithme additif a la programmation en nombres entiers et a la programmation non lineaire. C.R. Acad. Sci. Paris **258**(21), 5136 (1964)

[8] Beale, E.: Branch and bound methods for mathematical programming systems. In: Annals of Discrete Mathematics, vol. 5, pp. 201–219. Elsevier (1979)

[9] Beale, E., Forrest, J.J.: Global optimization using special ordered sets. Mathematical Programming **10**(1), 52–69 (1976)

[10] Beale, E., Tomlin, J.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. OR **69**(447-454), 99 (1970)

[11] Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods & Software **24**(4-5), 597–634 (2009)

[12] Benson, H.P.: A finite algorithm for concave minimization over a polyhedron. Naval Research Logistics Quarterly **32**(1), 165–177 (1985)

[13] Bertsimas, D., King, A., Mazumder, R.: Best subset selection via a modern optimization lens. The Annals of Statistics **44**(2), 813–852 (2016). URL `http://www.jstor.org/stable/43818629`

[14] Bretthauer, K.M., Shetty, B.: The nonlinear resource allocation problem. Operations research **43**(4), 670–683 (1995)

[15] Bretthauer, K.M., Shetty, B.: The nonlinear knapsack problem–algorithms and applications. European Journal of Operational Research **138**(3), 459–472 (2002)

[16] Bretthauer, K.M., Shetty, B.: A pegging algorithm for the nonlinear resource allocation problem. Computers & Operations Research **29**(5), 505–527 (2002)

[17] Bretthauer, K.M., Victor Cabot, A., Venkataramanan, M.: An algorithm and new penalties for concave integer minimization over a polyhedron. Naval Research Logistics (NRL) **41**(3), 435–454 (1994)

[18] Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science **17**(2), 97–106 (2012)

[19] Czyzyk, J., Mesnier, M.P., Moré, J.J.: The NEOS server. IEEE Journal on Computational Science and Engineering **5**(3), 68–75 (1998)

[20] Dolan, E.D.: The NEOS server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory (2001)

[21] D'Ambrosio, C., Lee, J., Wächter, A.: An algorithmic framework for MINLP with separable non-convexity. In: Mixed Integer Nonlinear Programming, pp. 315–347. Springer (2012)

[22] Floudas, C.A.: Deterministic global optimization: theory, methods and applications, vol. 37. Springer Science & Business Media (2013)

[23] Floudas, C.A., Gounaris, C.E.: A review of recent advances in global optimization. Journal of Global Optimization **45**(1), 3 (2009)

[24] Floudas, C.A., Visweswaran, V.: Quadratic optimization. In: Handbook of global optimization, pp. 217–269. Springer (1995)

[25] Fox, G.E., Baker, N.R., Bryant, J.L.: Economic models for r and d project selection in the presence of project interactions. Management science **30**(7), 890–902 (1984)

[26] Gallo, G., Hammer, P.L., Simeone, B.: Quadratic knapsack problems. In: Combinatorial optimization, pp. 132–149. Springer (1980)

[27] Geißler, B., Martin, A., Morsi, A., Schewe, L.: Using piecewise linear functions for solving MINLPs. In: Mixed integer nonlinear programming, pp. 287–314. Springer (2012)

[28] Gropp, W., Moré, J.J.: Optimization environments and the NEOS server. In: M.D. Buhman, A. Iserles (eds.) Approximation Theory and Optimization, pp. 167–182. Cambridge University Press (1997)

[29] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual. Version 9.1 (2020). URL `http://www.gurobi.com`

[30] Hanoch, G., Levy, H.: Efficient portfolio selection with quadratic and cubic utility. The Journal of Business **43**(2), 181–189 (1970)

[31] Henin, C., Doutriaux, J.: A specialization of the convex simplex method to cubic programming. Rivista di matematica per le scienze economiche e sociali **3**(2), 61–72 (1980)

[32] Horst, R., Tuy, H.: Global optimization: Deterministic approaches. Springer Science & Business Media (2013)

[33] Karuppiah, R., Grossmann, I.E.: A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. Journal of global optimization **41**(2), 163–186 (2008)

[34] Keha, A.B., de Farias Jr, I.R., Nemhauser, G.L.: Models for representing piecewise linear cost functions. Operations Research Letters **32**(1), 44–48 (2004)

[35] Keha, A.B., de Farias Jr, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Operations research **54**(5), 847–858 (2006)

[36] Kesavan, P., Allgor, R.J., Gatzke, E.P., Barton, P.I.: Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. Mathematical Programming **100**(3), 517–535 (2004)

[37] Knuth, D.E.: Johann Faulhaber and sums of powers. Mathematics of Computation **61**(203), 277–294 (1993)

[38] Land, A.H., Doig, A.G.: An automatic method for solving discrete programming problems. In: 50 Years of Integer Programming 1958-2008, pp. 105–132. Springer (2010)

[39] Lee, S., Grossmann, I.E.: A global optimization algorithm for nonconvex generalized disjunctive programming and applications to process systems. Computers & Chemical Engineering **25**(11-12), 1675–1697 (2001)

[40] Leyffer, S., Sartenaer, A., Wanufelle, E.: Branch-and-refine for mixed-integer nonconvex global optimization. Preprint ANL/MCS-P1547-0908, Mathematics and Computer Science Division, Argonne National Laboratory **39**, 40–78 (2008)

[41] Li, D., Sun, X.: Nonlinear integer programming, vol. 84. Springer Science & Business Media (2006)

[42] Li, H.L., Huang, Y.H., Fang, S.C.: A logarithmic method for reducing binary variables and inequality constraints in solving task assignment problems. INFORMS Journal on Computing **25**(4), 643–653 (2013)

[43] Li, H.L., Huang, Y.H., Fang, S.C.: Linear reformulation of polynomial discrete programming for fast computation. INFORMS Journal on Computing **29**(1), 108–122 (2017). DOI 10.1287/ijoc.2016.0716. URL `https://doi.org/10.1287/ijoc.2016.0716`

[44] Li, H.L., Lu, H.C.: Global optimization for generalized geometric programs with mixed free-sign variables. Operations research **57**(3), 701–713 (2009)

[45] Li, H.L., Lu, H.C., Huang, C.H., Hu, N.Z.: A superior representation method for piecewise linear functions. INFORMS Journal on Computing **21**(2), 314–321 (2009)

[46] Luss, H., Gupta, S.K.: Allocation of effort resources among competing activities. Operations Research **23**(2), 360–366 (1975)

[47] Maass, K.L., Lo, V.M.H., Weiss, A., Daskin, M.S.: Maximizing diversity in the engineering global leadership cultural families. Interfaces **45**(4), 293–304 (2015)

[48] MacMillan, K., Sondow, J.: Proofs of power sum and binomial coefficient congruences via Pascal's identity. The American Mathematical Monthly **118**(6), 549–551 (2011)

[49] Marsten, R.E., Morin, T.L.: A hybrid approach to discrete mathematical programming. Mathematical Programming **14**(1), 21–40 (1978)

[50] McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I—convex underestimating problems. Mathematical programming **10**(1), 147–175 (1976)

[51] Nielsen, S.S., Zenios, S.A.: A massively parallel algorithm for nonlinear stochastic network problems. Operations Research **41**(2), 319–337 (1993)

[52] Regucki, P., Lewkowicz, M., Krzyżyńska, R.: Optimization of thermal-flow processes in a system of conjugate cooling towers. Heat Transfer Engineering **41**(22), 1938–1948 (2020)

[53] Ryoo, H.S., Sahinidis, N.V.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering **19**(5), 551–566 (1995)

[54] Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. Journal of global optimization **8**(2), 107–138 (1996)

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

[56] Shetty, B., Muthukrishnan, R.: A parallel projection for the multicommodity network model. Journal of the Operational Research Society **41**(9), 837–842 (1990)

[57] Shimada, N., Yamazaki, N., Takano, Y.: Multi-objective optimization models for many-to-one matching problems. Journal of Information Processing **28**, 406–412 (2020). DOI 10.2197/ipsjjip.28.406

[58] Smith, E.M., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Computers & Chemical Engineering **21**, S791–S796 (1997)

[59] Tawarmalani, M., Sahinidis, N.V.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical programming **99**(3), 563–591 (2004)

[60] Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103**(2), 225–249 (2005)

[61] Tuy, H.: Concave programming under linear constraints. Soviet Math. **5**, 1437–1440 (1964)

[62] Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. Operations research **58**(2), 303–315 (2010)

[63] Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming **128**(1), 49–72 (2011)

[64] Weingartner, H.M.: Capital budgeting of interrelated projects: survey and synthesis. Management Science **12**(7), 485–516 (1966)

[65] Weisstein, E.W.: Bernoulli number. https://mathworld.wolfram.com/ (2002)

[66] Yıldız, S., Vielma, J.P.: Incremental and encoding formulations for mixed integer programming. Operations Research Letters **41**(6), 654–658 (2013)