Defining Core Problems for Set Covering Instances Using Machine Learning: a Proof of Concept

Sameh Al-Shihabi[1 [0000−0001−7634−756X]]

Industrial Engineering and Engineering Management Department, University of Sharjah, PO Box 27272, Sharjah, United Arab Emirates salshihabi@sharjah.ac.ae

Abstract. In this study, artificial intelligence (AI) and operations research (OR) are integrated to address the set covering problem (SCP) using the core concept (CP). A dataset of 45 small SCP instances with known optimal solutions was divided according to the 60–40 rule, with 27 instances used for training and 18 reserved for verification. The training set developed an XGBoost (XGB) model that estimates the probability of each variable taking the value 1 in the optimal solution. The model was trained with both classical CP features, such as reduced costs, and new features derived from linear programming (LP) relaxations. Eight candidate features were tested, and feature selection using Shapley additive explanation values reduced the set to three inputs: the LP solution value, the reduced cost, and a modified reduced cost that incorporates the LP solution value. In all experiments, the XGB model parameters were optimized through a structured grid search. Prediction probabilities from the trained model were then used to define two thresholds: variables above the upper threshold were fixed to 1, those below the lower threshold to 0, and the remaining variables formed a core problem, solved using *Gurobi*. With this approach, all 18 verification instances were solved to optimality. Furthermore, 18 of 20 large SCP instances- not used in XGB training- were solved to optimality.

Keywords: Machine learning · XGB · Set covering problem · SHAP values · Grid search

1 Introduction

Artificial intelligence (AI) and operations research (OR) are increasingly combined in decision systems [5,21]. A systematic classification of AI–OR integration is presented in [5], where three approaches are identified: *end-to-end* (AI solves the OR problem), *configuration* (AI supports the solution process), and *alongside* (AI interacts with an OR algorithm during execution). This paper introduces a new integration approach, termed *Learning-Based Core Identification (LBCI)*, in which AI is used to define the core problem (CP) of a binary integer program (BIP) [3,16,19], while an OR solver is used to optimally solve the resulting CP. The core concept partitions the variable set N into an adjunct set N^{adj} and a

core set N^{core}. Variables in N^{adj} are fixed to 0 (set N_0^{adj}) or 1 (set N_1^{adj}), forming a partial solution x^{adj}, while variables in N^{core} define the reduced BIP. If the fixed variables coincide with their values in the optimal BIP solution ${x^*}^{\text{BIP}}$, and ${x^*}^{\text{CP}}$ is the optimal CP solution, then $(x^{\text{adj}}, {x^*}^{\text{CP}}) = {x^*}^{\text{BIP}}$.

Figure 1 illustrates how an ML classifier identifies N_0^{adj} and N_1^{adj}, thereby defining the CP, which is then solved using an OR solver. The motivation for LBCI is aligned with the example in [5], where a delivery company repeatedly solves TSP instances in the Montreal area that share recurring structural patterns. LBCI exploits this setting by learning, from previously solved instances, which variables can be correctly fixed to define a reduced core problem that is easier to solve.

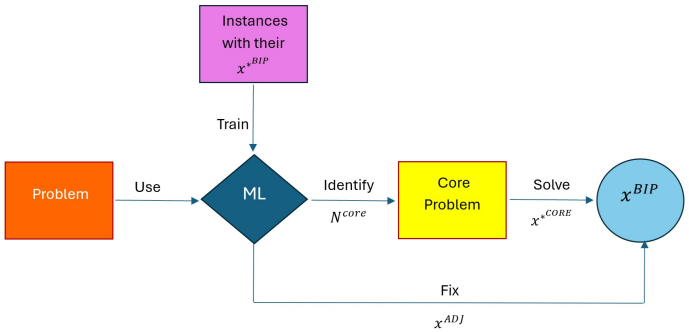

Fig. 1. Integration of the ANN-based variable classifier with an OR solver for solving binary integer programs via core problem identification.

The core concept requires defining: (i) an efficiency measure, τ, for classifying the variables and (ii) an approach to determine the size of N^{core}, alternatively, define the N_0^{adj} and N_1^{adj} sets. The efficiency measure for variable $j \in N$, τ_j, used by researchers simply compares the benefits to the costs of selecting a variable. For example, the first attempt to use the core concept was to solve the knapsack problem (KP) where $\tau_j = \frac{c_j}{a_j}$ is used as an efficiency measure [3]. Variables leading to a high increase in the objective function, i.e., high c_j, while consuming a small amount from the limiting resource, a_j, are preferred over other variables. Considering more than one constraint in the multidimenesional knapsack problem (MKP), $\tau_j = \frac{c_j}{\sum_{i=1}^{m} a_{ij} \times r_i}$ was used in [18], where r_i is the surrogate multiplier of constraint $i \in M$. e.g., the Lagrangian multiplier of the constraint. Surrogate multipliers were replaced by reduced costs, $\pi(s)$, in [20] when solving the MKP. Replacing division by subtraction in $\tau_j = \frac{c_j}{\sum_{i=1}^{m} a_{ij} \times r_i}$ leads to the reduced cost π_j formula, as shown in Equation 1, which was used in [1]. Both the sign and the magnitude of π_j are then used to assign variable

$j \in N$ to N_0^{adj}, N_1^{adj}, or N^{core}: large negative (positive) reduced costs imply $x_j = 1$ ($x_j = 0$) for minimization problems. Researchers compare the π_j values to selected threshold values to classify variables into the N^{adj} and N^{core} sets [1,2].

$$\tau_j = \pi_j = c_j - \sum_{i \in M} \mu_i \, a_{ij}. \tag{1}$$

Fixed-size cores were employed in [3] and [19]. Researchers, however, preferred variable-size cores that are compatible with the number of problem variables [16,20]. Since harder problem instances are expected to require larger core sizes, normalized reduced costs were adopted as efficiency measures in [12], because hard problems show less dispersion in normalized reduced costs compared to easy ones, which enabled enlarging N^{core} through threshold-based selection of variables in the resulting CP.

LBCI considers the segregation of variables into the N^{adj} and N^{core} sets as a classification problem. Thus, the efficiency measure, τ_j, represents a classification probability as shown in Equation 2, and segregation thresholds are defined based on the distribution of the classification probabilities. The values $\{\tau_j\}_{j=1}^{n}$ define an empirical distribution $\mathcal{F}_\tau$, which is used to derive the segregation thresholds.

$$\tau_j = Pr\{x_j^{*^{BIP}} = 1\} \tag{2}$$

To create an AI classifier to identify CP, a testbed of instances with known solutions is needed to train the AI model and evaluate its classification accuracy. For this reason, the set covering problem (SCP) is used in this experimental investigation [7]. The SCP is a classical problem that has been used to test several algorithms, such as genetic algorithms [4] and tabu search [8]. In addition, it provides benchmarking instances with known solutions that can be used to achieve the objective of this study. The SCP consists of selecting a subset of variables from N such that each constraint $i \in M$ is covered by at least one selected variable, as shown in Equation 4. The objective is to minimize the sum of costs associated with the selected variables, as shown in Equation 3. As a BIP, all variables in the SCP are binary, as shown in Equation 5, and the coefficient a_{ij} equals 1 if variable j covers constraint i, and 0 otherwise.

$$Minimize\ z = \sum_{j \in N} c_j x_j \tag{3}$$

s.t.

$$\sum_{j \in N} a_{ij} x_j \geq 1, \forall i \in M \tag{4}$$

$$x_j \in \{0, 1\}, \forall j \in N \tag{5}$$

The purpose of this research is not to introduce a novel, leading-edge solution methodology for a set of classical SCP benchmark set, but rather to illustrate the use of AI in classifying variables into the sets N_0^{adj}, N_1^{adj}, and N^{core}. The use of AI to classify variables opens the door to testing multiple features during training. Thus, as a by-product of this work, new features that have not been

used by researchers before are proposed, and their impact on the AI classifier is tested. The Extreme Gradient Boosting (XGB) algorithm [10] is adopted in this experiment for its ability to capture complex data patterns, resist overfitting, and offer both interpretability and strong predictive power [17].

The rest of this paper is organized as follows. Section 2 defines the components of the experiment behind the LBCI, followed by the methodology used to define thresholds in Section 3. Experiments are presented in Section 4, where small unseen instances are solved first then the scalability of the algorithm is checked by solving larger instances. Section 5 concludes the paper and suggests several future research directions.

2 Experiment components

In this section, the components of the XGB classifier are described. The section starts by describing the SCP benchmark instances used in this experiment, followed by a description of the XGB classifier and its configuration. Suggested features are then presented along with a description of how input features are selected.

2.1 SCP benchmarking sets

In this research, eleven well-known SCP benchmarking datasets are utilized. These benchmarking datasets can be downloaded from http://people.brunel.ac.uk/mastjjb/jeb/info.html, and their basic information are summarized in Table 1. Columns 1 and 6 of Table 1 indicate the set name. The names of the instances start with the set name, followed by the instance number. For example, instance scpa.1 is the first instance in set scpa. Columns 2 and 7 specify the number of instances within each set. Instances belonging to the same set share identical numbers of rows, columns 3 and 8, and variables, columns 4 and 9, of Table 1. Additionally, they exhibit the same density, representing the average number of columns covering any of the constraints, as indicated in columns 5 and 10. Instances from sets scp4–scpd can be optimally solved using the commercial solver *Gurobi* in a reasonable time. Consequently, these instances are selected for generating the data required to train the ML models.

The optimal IB and LP-relaxation solutions of 60% of the small instances, namely, the first six instances from sets scp4 and scp5 and the first three instances from sets scp6, scpa, scpb, scpc, and scpd, are used to train the ML models. For each variable in an instance, $x_j^{*^{BIP}}$ is used as a classification target.

2.2 XGB configuration

XGB is an ensemble learning method based on gradient-boosted decision trees, where multiple weak learners are sequentially combined to improve classification performance [10]. A structured Grid Search (GS) strategy [6,13] is employed to tune the XGB algorithm over a representative set of hyperparameters. This

Table 1. Description of the benchmarking sets used in this study. Instances from sets `scp4`–`scpd` are used to train the ML models.

set scp	Number of instances	Number of rows	Number of columns	Density %	set scp	Number of instances	Number of rows	Number of columns	Density %
4	10	200	1,000	2	c	5	400	4,000	2
5	10	200	2,000	2	d	5	400	4,000	5
6	5	200	1,000	5	NRE	5	500	5,000	10
a	5	300	3,000	2	NRF	5	500	5,000	20
b	5	300	3,000	5	NRG	5	1,000	10,000	2
					NRH	5	1,000	10,000	5

strategy enables a systematic exploration of the hyperparameter search space. Table 2 lists the hyperparameters and candidate values used to identify the best XGB configuration across the experiments.

Table 2. Hyperparameter search space used in the XGB grid search.

Parameter	Options
Maximum depth of trees	3, 6
Learning rate	0.01, 0.1
Number of estimators (trees)	100, 200
Subsample ratio	0.8, 1.0
Column sampling (per tree)	0.8, 1.0

The training process uses 3-fold cross-validation to identify the optimal configuration based on classification accuracy. The training data are highly imbalanced, with 1,523 variables belonging to class 1 ($x^{*^{\text{BIP}}} = 1$) and 61,477 variables belonging to class 0 ($x^{*^{\text{BIP}}} = 0$). To address this imbalance, the *scale_pos_weight* parameter is set to the ratio between the number of class 0 and class 1 samples when invoking the *XGBClassifier*.

2.3 Features

For each variable $j \in N$, we use c_j, x_j^{LP}, and π_j as inputs to the ANN prediction model. The values x_j^{LP} and π_j are obtained from the LP relaxation of the training instances, whereas c_j is part of the problem input. We also introduce several features not previously proposed in the literature (Table 3). Before describing these features, we define two terms:

1. $\text{cover}(i) := \{\, j \in N : a_{ij} = 1 \,\}$ is the set of columns that cover row i;
2. $\mathcal{B}_{\text{bind}} := \{\, i : \mu_i > 0 \,\}$ is the set of binding constraints at the LP solution.

The rationale behind the proposed features in Table 3 is simple. The binary indicators $B(x_j^{LP})$ and $B(\pi_j)$ are used to capture sign/presence information

Table 3. Suggested novel features used by the prediction models.

Feature	Description	Formula
Binary LP-solution indicator ($\mathbf{B(x_j^{LP})}$)	Indicator for $x_j^{LP} > 0$. **Example:** if $x_j^{LP} = 0.25$, then $B(x_j^{LP}) = 1$.	$B(x_j^{LP}) = 1$ if $x_j^{LP} > \varepsilon,\ 0$ otherwise
Binary reduced-cost indicator ($\mathbf{B(\pi_j)}$)	Indicator for $\pi_j > 0$. **Example:** if $\pi_j = 2.5$, then $B(\pi_j) = 1$.	$B(\pi_j) = 1$ if $\pi_j > \varepsilon,\ 0$ otherwise
Modified reduced cost ($\mathbf{M(\pi_j)}$)	Reduced-cost–like score including x_j^{LP} over binding rows. **Example:** $x_j^{LP} = 0.4$, $a_{1j} = a_{2j} = 1$, $\mu_1 = 2$, $\mu_2 = 3$ $\Rightarrow M(\pi_j) = 0.4(2+3) = 2.0$.	$M(\pi_j) = \sum_{i \in \mathcal{B}_{\text{bind}}} x_j^{LP}\, \mu_i\, a_{ij}$
Binding constraints covered ($\mathbf{BCC_j}$)	Number of binding constraints covered by j when $x_j^{LP} > 0$. **Example:** if $x_j^{LP} > 0$ and j covers 3 binding constraints, then $BCC_j = 3$.	$BCC_j = \lvert\{\, i \in \mathcal{B}_{\text{bind}} \mid j \in \text{cover}(i)\,\}\rvert$ if $x_j^{LP} > \varepsilon,\ 0$ otherwise
Variable replacement cost ($\mathbf{RP_j}$)	For $x_j^{LP} > 0$, sum over binding i covered by j of the extra cost from replacing j with the cheapest $k \neq j$ that also covers i and has $c_k \geq c_j$, weighted by μ_i; add 0 if none. **Example:** $c_j = 3$, $\mu_{i_1} = 3$, $\mu_{i_2} = 2$, cheapest qualifying alternatives: 5 (for i_1), 4 (for i_2) $\Rightarrow RP_j = (5-3)\cdot 3 + (4-3)\cdot 2 = 8$.	$RP_j = \sum_{i \in \mathcal{B}_{\text{bind}},\ j \in \text{cover}(i)} \left(\min_{\substack{k \in \text{cover}(i) \setminus \{j\} \\ c_k \geq c_j}} c_k - c_j \right) \mu_i$ if $x_j^{LP} > \varepsilon,\ 0$ otherwise

while ignoring magnitudes. The feature $M(\pi_j)$ augments the reduced-cost values by weighting row dual costs by the LP participation of j over binding rows. The count BCC_j reflects how many binding constraints j covers. Finally, RP_j measures the (dual-weighted) cost penalty of excluding a column with $x_j^{LP} > 0$.

The above features were computed for each variable in the selected instances. In total, eight features, in addition to x_{IP}^* values, were collected for 63,000 variables to construct the Pearson-correlation heatmap [11], shown in Figure 2. Figure 2 indicates that x^{LP} is highly correlated with $x^{*^{BIP}}$ ($\rho = 0.90$), whereas the binary indicator, $B(x^{LP})$, has $\rho = 0.78$. It was expected that the objective coefficient c to have a stronger correlation with $x^{*^{BIP}}$ than the observed $\rho = 0.25$. Interestingly, π has a small correlation with $x^{*^{BIP}}$ ($\rho = -0.25$), while its indicator and modified versions, $B(\pi)$ and $M(\pi)$, show higher correlations of -0.72 and 0.58, respectively. The suggested BCC and RP features exhibit very low correlations with the IP solution; however, they are also weakly correlated with the other features, which may make them informative by adding non-redundant signal.

Correlation coefficients are global measures because they aggregate information over all data points; moreover, Pearson correlation does not capture nonlinear relationships [9]. Therefore, we use Shapley Additive exPlanations (SHAP) to select the input features [15]. SHAP computes, for each observation and feature, a value that quantifies the feature's marginal contribution to the model's prediction; aggregating these values yields feature importance for the classification model.

SHAP values were computed with the *SHAP* library in Python using *TreeExplainer* for the trained XGB model. The dataset was standardized with a fitted *StandardScaler* to match model training, and the feature set comprised all columns. For each feature, the mean absolute SHAP value was calculated

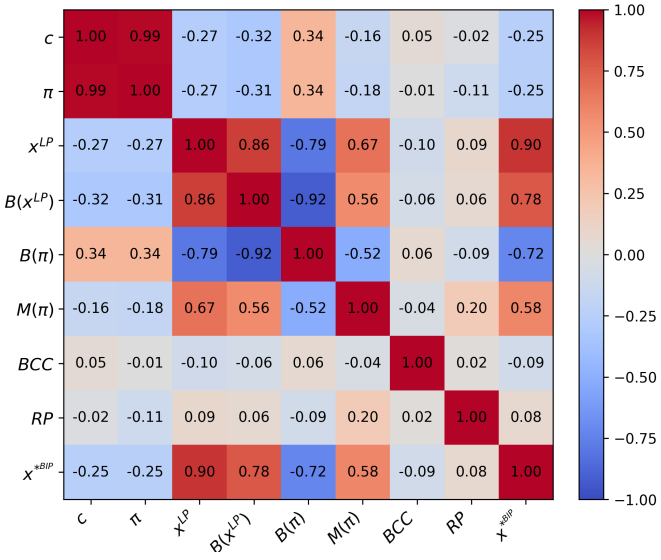

Fig. 2. Correlation heatmap for the eight features and x^{*BIP}.

and used to rank features in descending order. A backward feature-elimination procedure was then applied: the feature with the smallest mean absolute SHAP value was removed, the model was retrained on the remaining features, and the classification accuracy was recorded. The procedure was repeated while accuracy improved and terminated once accuracy decreased.

Figure 3 shows the SHAP values for the XGB model. The top two features by mean absolute SHAP value are x^{LP} and π, followed by $\mathbf{M}(\pi)$. Guided by these rankings, the backward feature elimination retained only these three features. The XGB model achieved the lowest misclassification rate with this three-feature subset (Table 4). Accordingly, the selected per-variable features are x_j^{LP}, π_j, and $M(\pi_j)$.

Table 4. XGB accuracy after SHAP-based feature elimination.

Removed Features	XGB
None	376
RP	373
RP & *BCC*	375
RP & *BCC* & $B(\pi)$	373
RP & *BCC* & $B(\pi)$ & $B(x)$	375
RP & *BCC* & $B(\pi)$ & $B(x)$ & c	372
RP & *BCC* & $B(\pi)$ & $B(x)$ &c & $M(\pi)$	387

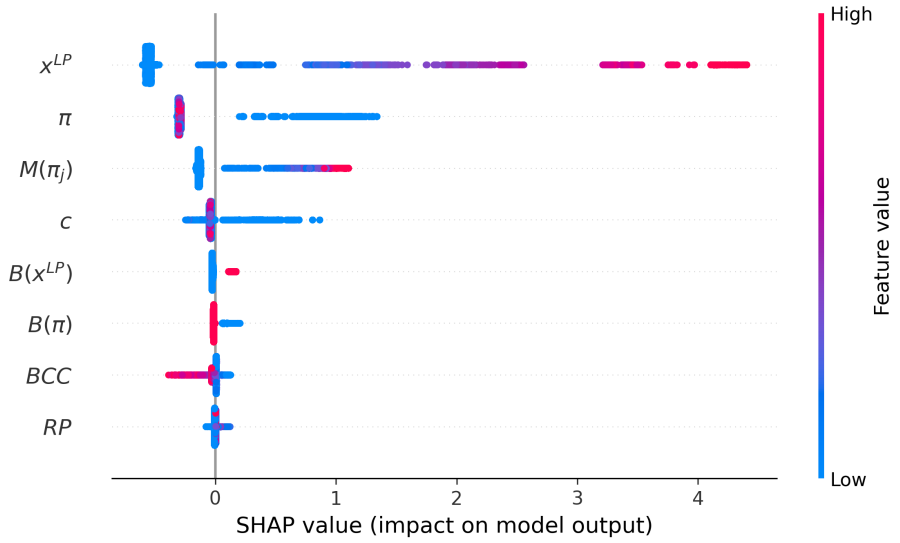

Fig. 3. SHAP absolute values for the XGB model.

Although not reported, a feedforward artificial neural network (ANN) [14] was also used to train a prediction model similar to XGB. For ANN, all the features, except for c and π, were used by the best ANN model. Thus, the suggested features, despite not having a straightforward mathematical interpretation, can help the different ML classifiers. The ANN results were inferior to those of XGB; therefore, they are not presented in this work.

3 CP defining thresholds

The ML model generates a probability score $\tau_j = \Pr\left(x_j^{*^{BIP}} = 1\right)$ for every $j \in N$. As illustrated by the hypothetical distribution of τ_j values in Figure 4, two thresholds τ_0 and τ_1 $(0 \leq \tau_0 \leq \tau_1 \leq 1)$ are selected to define the CP sets. In the figure, the τ_j values are shown along a line, with variables having $x_j^{*^{BIP}} = 1$ highlighted in red. Variables with $\tau_j \leq \tau_0$ are assigned to N_0^{adj}, those with $\tau_j \geq \tau_1$ to N_1^{adj}, and those with $\tau_0 < \tau_j < \tau_1$ to N^{core}. The hypothetical plot also indicates potential misclassifications, e.g., blue circles (with $x_j^{*^{\text{BIP}}} = 0$) belonging to N_1^{adj}.

Figure 5 shows a sample histogram of the τ_j values for instance scp6.1. Due to the large number of columns compared to rows, it is clearly evident that the biggest cluster of τ_j values has probabilities approaching 0, as shown by the long column to the left of Figure 5. The second cluster, which corresponds to variables expected to have $x^{*^{BIP}} = 1$, has probabilities approaching 1, as demonstrated

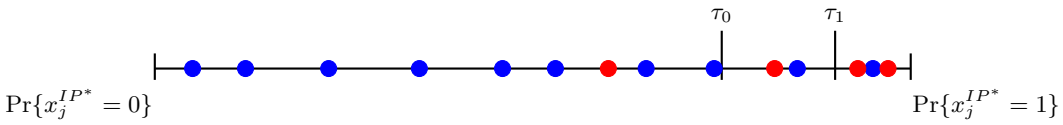

Fig. 4. Illustration of CP definitions using threshold values.

by the small column to the right of Figure 5. Figure 5 shows how the τ_0 and τ_1 thresholds are used to define N_0^{adj}, N_1^{adj}, and N^{core}.

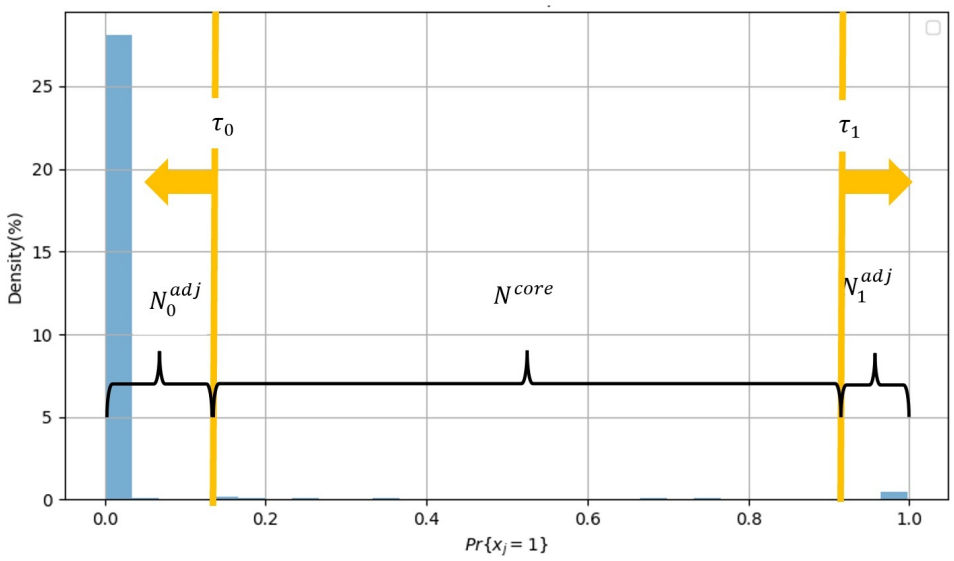

Fig. 5. Distribution of τ_j values for instance scp6.1 as found by the XGB model. Thresholds τ_0 and τ_1 define N_0^{adj}, N_1^{adj}, and N^{core}.

Unlike prior techniques that apply the same thresholds across instances to classify variables, this work employs an online, instance-specific procedure to determine τ_0 and τ_1. The procedure proceeds as follows: first, it partitions variables by their predicted probabilities into $G_0 = \{j : \tau_j \leq 0.5\}$ and $G_1 = \{j : \tau_j > 0.5\}$, Figure 5. Second, to mitigate boundary effects near 0.5, it trims the central 25% in each group—discarding the upper quartile by τ_j in G_0 and the lower quartile in G_1. Third, it fits a normal distribution to the remaining τ_j in each group, yielding (μ_0, σ_0) for G_0 and (μ_1, σ_1) for G_1, where μ and σ denote the mean and standard deviation. Finally, it defines τ_0 and τ_1 from these fitted normal models.

Table 5 reports the characteristics of the generated subsets N_0^{adj} and N_1^{adj} for the XGB predictions. The left half pertains to N_0^{adj} (varying τ_0) and the right half to N_1^{adj} (varying τ_1). For each subset, four threshold settings are evaluated: for τ_0, $\{\mu_0,\ \mu_0+\sigma_0,\ \mu_0+2\sigma_0,\ \mu_0+3\sigma_0\}$; and for τ_1, $\{\mu_1-3\sigma_1,\ \mu_1-2\sigma_1,\ \mu_1-\sigma_1,\ \mu_1\}$. Each row begins with the set name and two metrics. The first, $\mathbb{E}$[Size], is the average number of variables included in the corresponding subset over all instances in the set, e.g., the six instances of set 4, or the three instances of set a. The second, $\mathbb{E}$[Wrong], measures misclassifications. The formula used to calculate the average misclassified elements in N_0^{adj} is shown by Equation 6. An analogous expression is used for N_1^{adj} with the indicator $\mathbb{I}(\Pr(x_j^{\text{pred}}=1) > \tau_1 \mid x_j^{*\text{BIP}}=0)$.

$$\mathbb{E}\left[\sum_{j\in N_0^{\text{adj}}} \mathbb{I}\Big(\Pr(x_j^{\text{pred}}=1) \le \tau_0 \,\Big|\, x_j^{*\text{BIP}}=1\Big)\right] \tag{6}$$

Table 5. Expected size and misclassified variables of N_0^{adj} and N_1^{adj} under several threshold values.

		τ_0				τ_1			
Set	**Metric**	μ_0	$\mu_0+\sigma_0$	$\mu_0+2\sigma_0$	$\mu_0+3\sigma_0$	$\mu_1-3\sigma_1$	$\mu_1-2\sigma_1$	$\mu_1-\sigma_1$	μ_1
4	E[Size]	887.33	887.33	887.33	887.33	47.33	43.83	41.17	37.67
	E[W]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	E[Size]	1874.50	1874.50	1874.50	1874.50	50.33	43.50	39.00	37.33
	E[W]	0.00	0.00	0.00	0.00	1.67	1.00	0.50	0.33
6	E[Size]	910.67	910.67	910.67	910.67	25.00	21.67	20.67	8.33
	E[W]	0.00	0.00	0.00	0.00	2.00	1.33	0.67	0.00
a	E[Size]	2821.33	2821.33	2821.33	2821.33	54.00	48.33	40.67	24.33
	E[W]	0.00	0.00	0.00	0.00	5.00	3.67	1.33	0.33
b	E[Size]	2864.33	2864.33	2864.33	2864.33	35.00	29.67	22.33	15.33
	E[W]	0.00	0.00	0.00	0.00	5.00	3.33	1.00	0.33
c	E[Size]	3791.33	3791.33	3791.33	3791.33	61.00	52.67	47.33	26.67
	E[W]	0.00	0.00	0.00	0.00	6.00	3.67	2.00	0.00
d	E[Size]	3838.67	3838.67	3838.67	3838.67	33.67	27.00	21.33	17.67
	E[W]	0.00	0.00	0.00	0.00	6.00	4.00	2.00	1.00

Overall, Table 5 shows a clear asymmetry between the two adjunct sets and a stable behavior on the 0-side. For N_0^{adj}, $\mathbb{E}$[Size] and $\mathbb{E}$[Wrong] are identical across all four choices of τ_0, and $\mathbb{E}$[Wrong] $= 0$ for every set, indicating that, after trimming, the distribution in G_0 is extremely concentrated, effectively $\sigma_0 \approx 0$, and any $\mu_0 + k\sigma_0$ cutoff yields the same subset without errors. For N_1^{adj}, increasing τ_1 from $\mu_1 - 3\sigma_1$ to μ_1 steadily reduces $\mathbb{E}$[Size] and typically lowers $\mathbb{E}$[Wrong], often to zero, because the stricter cutoff admits only high-confidence variables, reduces false positives, and shifts borderline cases into N^{core} to be resolved by the CP. The gap in $\mathbb{E}$[Size] is large, hundreds for N_0^{adj} and only tens for N_1^{adj}, which is consistent with the sparsity of SCP solutions and with the histogram in Figure 5. Practically, these results suggest that τ_0 can be set at any of the four tested values with no impact. In contrast, a stricter τ_1, for example μ_1 or $\mu_1 - \sigma_1$, is preferable when prioritizing accuracy or a larger N^{core}.

4 Experiments

Two sets of experiments are conducted in this section. The remaining instances of sets 4–6 that were used to develop the LBCI algorithm are solved using different threshold values in the first experiment. The second experiment solves large instances that were not used in training the ML models. The second experiment is intended to understand the effect of scalability.

4.1 Small instances

In this experiment, instances from the same benchmarking set that were not used to train the XGB classifier are solved using the proposed algorithm. Table 6 reports, for each instance, the optimum solution, and the solution returned by the algorithm together with the resulting core size $|N^{\text{core}}|$ for four settings of k (0, 1, 2, and 3). An underlined objective value indicates that the solution equals the optimal solution for that instance.

Table 6. Results on instances of sets scp4–scpD not used in training, for $k \in \{0, 1, 2, 3\}$.

scp	**Opt**	$k = 0$		$k = 1$		$k = 2$		$k = 3$	
		Obj.	$\lvert N^{\text{core}}\rvert$	Obj.	$\lvert N^{\text{core}}\rvert$	Obj.	$\lvert N^{\text{core}}\rvert$	Obj.	$\lvert N^{\text{core}}\rvert$
4.7	430	<u>430</u>	84	<u>430</u>	73	<u>430</u>	59	<u>430</u>	49
4.8	492	<u>492</u>	92	<u>492</u>	70	<u>492</u>	63	<u>492</u>	63
4.9	641	<u>641</u>	90	<u>641</u>	90	<u>641</u>	73	<u>641</u>	67
4.10	514	<u>514</u>	85	<u>514</u>	57	<u>514</u>	50	<u>514</u>	42
5.7	293	<u>293</u>	109	<u>293</u>	88	<u>293</u>	78	<u>293</u>	70
5.8	288	<u>288</u>	98	<u>288</u>	98	<u>288</u>	98	<u>288</u>	94
5.9	279	<u>279</u>	109	<u>279</u>	79	<u>279</u>	66	<u>279</u>	66
5.10	265	<u>265</u>	112	<u>265</u>	69	<u>265</u>	56	<u>265</u>	56
6.4	131	<u>131</u>	59	<u>131</u>	59	<u>131</u>	59	<u>131</u>	56
6.5	161	<u>161</u>	78	162	71	68	66	168	62
A.4	234	<u>234</u>	143	<u>234</u>	142	235	133	236	124
A.5*	236	237	135	237	135	237	135	237	135
B.4	79	<u>79</u>	116	80	113	81	110	83	105
B.5	72	<u>72</u>	109	<u>72</u>	109	<u>72</u>	104	74	98
C.4	219	<u>210</u>	184	221	166	225	154	235	140
C.5	215	<u>215</u>	177	216	149	216	149	222	140
D.4	62	<u>62</u>	146	64	139	65	128	67	126
D.5	61	<u>61</u>	141	<u>61</u>	139	62	127	65	123

* An optimal solution was found for $k = -0.5$, leading to $|N^{\text{core}}| = 182$.

Table 6 shows that $k = 0$ yields the highest number of optimal solutions: 17/18 instances at $k = 0$, compared with 12/18 at $k = 1$, 10/18 at $k = 2$, and

9/18 at $k = 3$. The only instance not solved at $k = 0$ is A.4; after investigating negative k values, the optimum was reached at $k = -0.5$ with $|N^{\text{core}}| = 182$. Despite the negative k value, the increase in N^{core} was not significant.

4.2 Large instances

In this experiment, large SCP instances that were not used to develop the XGB classifier are solved. These instances have substantially more rows and columns than those used to train the ML model (Table 1). Moreover, the densities of the sets scpNRE and scpNRF are 10% and 20%, respectively, whereas the training instances did not exceed 5%. Each problem is solved twice: a two-sided reduction with $\tau_0 = \mu_0 + \sigma_0$ and $\tau_1 = \mu_1 - \sigma_1$, and a one-sided reduction with $\tau_0 = \mu_0 + \sigma_0$ and $\tau_1 = 1.0$, i.e., $N_1^{\text{adj}} = \emptyset$. The termination criterion was either to reach the best known value (BKV) or to allow *Gurobi* up to one hour on an Intel Core i5 125H @ 3.60 GHz.

Table 7 lists, for each instance, the BKV, the solution under the two-sided setting—objective value, $|N_0^{\text{adj}}|$, $|N^{\text{core}}|$, $|N_1^{\text{adj}}|$, and time—and the solution under the one-sided setting—objective value and time. The results indicate that XGB is highly reliable on the 0-side: with $\tau_1 = 1.0$, the BKV is attained in 18 of 20 instances, compared with 8 of 20 under the two-sided reduction. This suggests very few misclassifications into N_0^{adj}, except for scpNRH.2 and scpNRH.3, whereas the two-sided setting introduces more than ten misclassified variables into N_1^{adj} in several cases. Finally, four scpNRF instances have $N_1^{\text{adj}} = \emptyset$, which show that all variables have $\tau_j < 0.5$.

5 Conclusion

This study links AI and OR by using AI to develop a CP for a BIP, where the CP is solved using commercial software, *Gurobi*. Benchmark SCP instances with known optimal solutions are used to train an XGB prediction model. Eight input features are derived from the instances' input parameters and their LP-relaxation solutions, of which five features have never been used earlier and do not have a mathematical usage in OR. Using the SCP instances and their solutions, the ML model estimates each variable's probability of being 1 in the optimal BIP solution. A GS algorithm is used to tune the XGB model, and a backward-elimination approach based on SHAP values is used to choose proper input features. The XGB model only needed three input features, namely, LP-solution, reduced cost, and a modified reduced cost that considers the LP-solution value into account.

The set of core variables is determined by statistical cut-offs that are calculated from the prediction probability scores. The adjunct variables, which are not included in the core set, are fixed at either 0 or 1. The adjunct variables' values are combined with the CP solution to form the complete BIP solution, which can be considered as a matheuristic to solve the SCP. The generated matheuristic was capable of finding all the optimal solutions of instances having the same

Table 7. XGB-based solutions using two threshold settings for large instances.

scp	**BKV**	$\tau_0 = \mu_0 + \sigma_0,\ \tau_1 = \mu_1 - \sigma_1$					$\tau_0 = \mu_0 + \sigma_0,\ \tau_1 = 1.0$	
		Obj.	$\lvert N_0^{\text{adj}}\rvert$	$\lvert N^{\text{core}}\rvert$	$\lvert N_1^{\text{adj}}\rvert$	**Time (s)**	**Obj.**	**Time (s)**
NRE.1	29	29	4,838	154	8	12.1	29	17.3
NRE.2	30	31	4,828	170	2	21.7	30	36.8
NRE.3	27	28	4,832	155	13	14.9	27	23.9
NRE.4	28	29	4,848	142	10	29.0	28	20.5
NRE.5	28	28	4,837	141	12	12.4	28	30.7
NRF.1	14	14	4,857	143	0	23.5	14	23.5
NRF.2	15	15	4,851	149	0	23.3	15	23.3
NRF.3	14	14	4,850	148	2	31.6	14	47.2
NRF.4	14	14	4,855	145	0	24.6	14	24.6
NRF.5	13	13	4,855	145	0	36.6	13	26.6
NRG.1	176	177	9,571	379	50	20.6	176	57.8
NRG.2	154	157	9,562	389	49	20.6	154	141.8
NRG.3	166	168	9,589	360	51	21.1	166	323.7
NRG.4	168	172	9,586	367	47	22.7	168	36.5
NRG.5	168	170	9,574	381	38	23.4	168	53.6
NRH.1	63	63	9,649	341	10	52.0	63	381.2
NRH.2	63	64	9,647	345	10	3600.0	64	3600.0
NRH.3	59	60	9,660	328	12	3600.0	60	3600.0
NRH.4	58	58	9,660	333	7	203.0	58	52.8
NRH.5	55	56	9,659	327	14	196.4	55	58.2

characteristics of the instances used to train the XGB model. The matheuristic also proved adaptable because it stayed competitive on larger, harder instances than those used for training.

Several extensions could build on this research: exploring alternative ML algorithms, designing new features, applying the approach to problems beyond the SCP, or post-processing solutions obtained with this matheuristic using metaheuristics or heuristics. Since the current model scales well, an important direction is tackling large instances by first generating smaller representations of them to train an ML model, which can then be used to reduce their size. Finally, metaheuristics may exploit the probability scores to guide the search toward high-impact variables.

References

1. Al-Shihabi, S.: A hybrid of max–min ant system and linear programming for the k-covering problem. Computers & Operations Research **76**, 1–11 (2016). https://doi.org/https://doi.org/10.1016/j.cor.2016.06.006
2. Al-Shihabi, S.: Ants for identifying and optimizing the core set k-covering problem. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2552–2558. IEEE (2019)

3. Balas, E., Zemel, E.: An algorithm for large zero-one knapsack problems. operations Research **28**(5), 1130–1154 (1980). https://doi.org/https://doi.org/10.1287/opre.28.5.1130
4. Beasley, J.E., Chu, P.C.: A genetic algorithm for the set covering problem. European Journal of Operational Research **94**(2), 392–404 (1996). https://doi.org/10.1016/0377-2217(95)00357-6
5. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research **290**(2), 405–421 (2021). https://doi.org/10.1016/j.ejor.2020.06.044
6. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012). https://doi.org/10.5555/2188385.2188405
7. Caprara, A., Toth, P., Fischetti, M.: Algorithms for the set covering problem. Annals of Operations Research **98**(1), 353–371 (2000). https://doi.org/10.1023/A:1019262515027
8. Caserta, M.: Tabu search-based metaheuristic algorithm for large-scale set covering problems. Metaheuristics: Progress in Complex Systems Optimization pp. 43–63 (2007). https://doi.org/10.1007/978-0-387-71920-7_3
9. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & electrical engineering **40**(1), 16–28 (2014). https://doi.org/10.1016/j.compeleceng.2013.11.024
10. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794 (2016)
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research **3**(Mar), 1157–1182 (2003). https://doi.org/10.5555/944919.944968
12. Hill, R.R., Cho, Y.K., Moore, J.T.: Problem reduction heuristic for the 0–1 multidimensional knapsack problem. Computers & Operations Research **39**(1), 19–26 (2012). https://doi.org/https://doi.org/10.1016/j.cor.2010.06.009
13. Injadat, M., Moubayed, A., Nassif, A.B., Shami, A.: Systematic ensemble model selection approach for educational data mining. Knowledge-Based Systems **200**, 105992 (2020). https://doi.org/10.1016/j.knosys.2020.105992
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
15. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems **30** (2017). https://doi.org/10.48550/arXiv.1705.07874
16. Martello, S., Toth, P.: A new algorithm for the 0-1 knapsack problem. Management Science **34**(5), 633–644 (1988). https://doi.org/https://doi.org/10.1287/mnsc.34.5.633
17. Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. Frontiers in Neurorobotics **7**, 21 (2013). https://doi.org/10.3389/fnbot.2013.00021
18. Pirkul, H.: A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Research Logistics **34**(2), 161–172 (1987). https://doi.org/10.1002/nav.3800340206
19. Pisinger, D.: Core problems in knapsack algorithms. Operations Research **47**(4), 570–575 (1999). https://doi.org/https://doi.org/10.1287/opre.47.4.570
20. Puchinger, J., Raidl, G.R., Pferschy, U.: The core concept for the multidimensional knapsack problem. In: European Conference on Evolutionary Computation in Combinatorial Optimization. pp. 195–208. Springer (2006)

21. Wiberg, H., Dai, T., Lam, H., Kulkarni, R.: Synergizing artificial intelligence and operations research: Perspectives from informs fellows on the next frontier. INFORMS Journal on Data Science (2025). https://doi.org/https://doi.org/10.1287/ijds.2025.0077