

Equivalence-Based Reduction and Incremental Optimization for Categorical and Ordinal Classification

Yueh Lin^{*} Luis Fernando Pérez Armas[†] Stefano Nasini[‡] Martine Labbé[§]

September 17, 2026

Abstract

Exact mathematical programming formulations for classification tasks provide optimality guarantees, but their size typically grows exponentially with the number of predictive features. This paper shows that high-quality classifiers can be obtained through simple truncation and warm-start strategies that are directly compatible with state-of-the-art disjunctive normal form and decision tree representations. We first develop MILP and QUBO formulations for categorical and ordinal Boolean classification, establish equivalence conditions among alternative treatments of empty and contradictory clauses, and derive reduced saturated representations that lower the effective dimensionality of the clause space. Building on these structural results, we propose an incremental learning procedure that solves a sequence of smaller optimization problems, progressively enlarging the admissible model class while carrying forward high-quality incumbent solutions. We explore the advantage of quantum adiabatic/variational methods for producing warm-start solutions. Computational experiments on benchmark categorical datasets show that this approach substantially improves the solvability of large-scale categorical and ordinal Boolean classification problems.

Key words: Statistical classification, Boolean rules, Binary and quantum optimization, Incremental learning.

^{*}IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM- Lille Economie Management, F-59000 Lille, France, y.lin@ieseg.fr

[†]IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM- Lille Economie Management, F-59000 Lille, France, l.perezarmas@ieseg.fr

[‡]IESEG School of Management, Univ. Lille, CNRS, UMR 9221 - LEM- Lille Economie Management, F-59000 Lille, France, s.nasini@ieseg.fr

[§]Free University of Bruxelles, CP212, boulevard du Triomphe, 1050 Bruxelles, Belgium

1 Introduction

Classification is a fundamental problem in data analysis and machine learning, with applications ranging from medicine and engineering to economics and the social sciences (Hammer and Bonates 2006, Schneeweiss 2012). Given a collection of F features, the task is to partition the corresponding F -dimensional feature space into C classes, so as to match the observed feature–class associations provided by a training dataset. In many applications, classes are purely categorical, while in others they must respect a pre-specified ranking among labels. Decision trees (DTs) and Disjunctive Normal Forms (DNFs) are widely used representations for categorical and ordinal classification tasks (Loh 2014). Their popularity stems from their ability to translate statistical associations into simple logical rules: each path from root to leaf in a DT, or each clause in a DNF formula, corresponds to an interpretable rule that partitions the feature space, as studied since the work of Valiant (1984). Attempts to connect the DT and DNF paradigms date back to Pagallo (1989), who improved DNF learning by exploiting equivalence with DTs, and continue in Liu (2022), who induces DTs by solving an optimal Boolean rule subproblem at each node.

This said, finding optimal classifiers by within-sample error minimization is NP-hard in general (Karp 2009).¹ Constructive heuristics therefore provide approximate classifiers and feasible solutions that serve as strong primal bounds for subsequent exact optimization. Because optimality guarantees are important in healthcare and engineering contexts (Bertsimas and Dunn 2017, Hu et al. 2019), research has addressed exact solvability to support interpretability and predictive performance. Early contributions include Bertsimas and Shioda (2007), who partition the feature space into polyhedral regions by mixed-integer optimization, and Üney and Türkay (2006), who address the multiclass case. In the same vein, Chen and Lee (2018) use mixed-integer optimization for best-subset binary prediction, maximizing empirical classification accuracy over linear threshold rules. For DTs, Bertsimas and Dunn (2017) propose a mixed-integer formulation and report out-of-sample accuracy improvements of 1–2% in univariate and 3–5% in multivariate tasks, relative to the CART algorithm, the state-of-the-art greedy heuristic method widely available in modern statistical software (Loh 2014), with even larger gains in high-dimensional settings. Subsequent research has expanded this frontier via stronger formulations and decomposition-based algorithms (Aghaei et al. 2025, Verwer and Zhang 2017, 2019, Zhu et al. 2020), while a parallel line of work replaces discrete branching rules by randomized ones, trainable by continuous nonlinear optimization (Blanquero et al. 2020, 2021). Alongside these developments, a line of work has relied on SAT representations of tree structures (Narodytska et al. 2018, Janota and Morgado 2020), while others have learned globally optimal DTs

¹Beyond within-sample error minimization, trees are also optimized as decision objects encoding interpretable policies (Mišić 2020, Ciocan and Mišić 2022).

by exact discrete-search methods (Nijssen and Fromont 2010, Hu et al. 2019, Verhaeghe et al. 2020). From a different perspective, Clark and Niblett (1989), Marchand and Shawe-Taylor (2002), Boros et al. (2011), Dash et al. (2018) and Lawless et al. (2023a) have focused on DNF representations, addressing their optimization either via column generation (Dash et al. 2018) or by equivalence-based reduction to set-covering (Marchand and Shawe-Taylor 2002, Clark and Niblett 1989). While exact formulations provide optimality certificates, their size grows rapidly with model complexity. In the Mixed Integer Linear Programming (MILP) models of Bertsimas and Dunn (2017) and Aghaei et al. (2025), the number of variables grows exponentially with the tree depth, while in the DNF formulation of Dash et al. (2018) the number of candidate clauses grows combinatorially with F . With a view to improving scalability in F , this paper develops an equivalence-based reduction framework and contributes along three directions. First, we develop MILP and Quadratic Unconstrained Binary Optimization (QUBO) formulations covering both categorical classification, based on unique class assignment, and ordinal classification, based on a first-true logic under a pre-specified class order.² Within this framework we uncover equivalence conditions among alternative treatments of empty and contradictory clauses, and establish a reduced formulation via a saturation principle: any DNF with clauses of at most D literals admits an equivalent saturated representation whose non-false clauses contain exactly D literals. This shrinks the effective clause dictionary and provides a common space into which depth- D DTs embed. Second, we tackle QUBO instances by quantum adiabatic/variational procedures adapted from Kadowaki and Nishimori (1998) and Albash and Lidar (2018), which can be used to generate high-quality warm-start solutions for the subsequent MILP optimization. Third, we solve the reduced formulations through a sequence of smaller problems of increasing expressiveness, in the spirit of stepwise regression but embedded, at each given iteration, within an endogenous features selection framework. These iterations naturally produce warm-start solutions for the next ones, since solutions obtained at lower complexity levels can be lifted to initialize larger formulations. This warm-start-centric perspective is particularly relevant for DNF-based formulations, because their clause budget can be set independently of the literal depth, unlike DT formulations, where the number of leaves is tied to the depth.

On the numerical side, we benchmark both classical and quantum optimization methods for solving the reduced problems. The computational results show that the proposed incremental strategy is effective precisely because it is built around reusing, lifting, and improving warm-start solutions. Across a wide range of benchmark datasets, the resulting pipeline consistently attains optimal or near-optimal classification accuracy in both DNF and DT representations. Furthermore, our

²State-of-the-art formulations primarily target categorical data. The DT approaches in Bertsimas and Dunn (2017) and Aghaei et al. (2025) support multiclass problems ($C > 2$), whereas the DNF formulation in Dash et al. (2018) is restricted to binary classification ($C = 2$).

investigation of quantum adiabatic/variational algorithms shows that these methods can deliver highly competitive classifiers, although their performance remains dependent on hardware limitations. The paper is organized as follows. Section 2 introduces the model setup. Section 3 presents the MILP and QUBO formulations and the associated equivalence results. Section 4 describes the incremental learning procedures. Section 5 reports the computational experiments. Section 6 concludes. Proofs and auxiliary material are available in the online appendix.

2 Baseline Boolean model

We consider a set of individual observations $\mathcal{I}$ (with $|\mathcal{I}| = I \geq 1$) and a set of classes $\mathcal{C}$ (with $|\mathcal{C}| = C \geq 2$) into which observations can be partitioned. Given F categorical features, a training dataset is given by a list of pairs $\{(\mathbf{v}_i, u_i)\}_{i \in \mathcal{I}}$, where $\mathbf{v}_i \in \{0, 1\}^F$ is the F -dimensional vector of individual i 's features and $u_i \in \mathcal{C}$ is the class to which individual i belongs. We introduce the binary vector $\mathbf{u}_i = [u_{i,1}, \dots, u_{i,C}]^\top$, where $u_{i,c} = 1$ if individual i belongs to class c , and $u_{i,c} = 0$ otherwise. In Boolean classification, $\mathcal{I}$ is to be partitioned into C classes based solely on logical expressions involving the individual features. These can be represented as OR-of-ANDs expressions (referred to as DNF) defined over a collection $\mathcal{T}$ of T conjunctions. Formally, in Boolean classification tasks, this individual-class assignment boils down to the search of a multivalued function of the form:

$$c \in U(\phi, \mathbf{v}) \iff \bigvee_{t \in \mathcal{T}} \left(\left(\bigwedge_{f: \bar{\phi}_{f,c}^t = 1} v_f \right) \wedge \left(\bigwedge_{f: \underline{\phi}_{f,c}^t = 1} \neg v_f \right) \right), \quad (1)$$

where $\bar{\phi}_{f,c}^t$ and $\underline{\phi}_{f,c}^t$ denote binary indicators of whether feature $f \in \mathcal{F}$ is affirmed ($\bar{\phi}_{f,c}^t = 1$) or negated ($\underline{\phi}_{f,c}^t = 1$) in the t -th conjunction of class $c \in \mathcal{C}$, respectively. Throughout the paper, vectorizations of arrays along a given dimension are denoted by a dot in the corresponding index. Consistently, we write $\bar{\phi}_{\cdot,c}^t = (\bar{\phi}_{1,c}^t, \dots, \bar{\phi}_{F,c}^t) \in \{0, 1\}^F$ and $\underline{\phi}_{\cdot,c}^t = (\underline{\phi}_{1,c}^t, \dots, \underline{\phi}_{F,c}^t) \in \{0, 1\}^F$. We also use the shorter notation $\phi_{f,c}^t$ to refer to $\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t$ when the context allows.

The key distinction between categorical and ordinal classification reduces to the image set of $U(\phi, \mathbf{v})$, where $U(\phi, \mathbf{v}) \in 2^{\mathcal{C}}$ for Ordinal Classification (OC), and $U(\phi, \mathbf{v}) \in \mathcal{C}$ for Categorical Classification (CC). This is equivalent to imposing a constraint enforcing that exactly one DNF formula evaluates to true for any individual data point in the case of CC. Conversely, for OC, several DNFs corresponding to classes $c_1, \dots, c_C$ may evaluate to true, and a hierarchy is established among classes. In this case, the first class whose DNF evaluates to true is defined as the classification result for the data point, where the ordering of classes is determined beforehand. To formalize this distinction, we consider an algebraic characterization in terms of the binary indicators $\bar{\phi}_{f,c}^t$ and $\underline{\phi}_{f,c}^t$.

Definition 1 (DNF). For each class $c \in \mathcal{C}$, we define the set of DNFs involving T (empty and nonempty) clauses and F features as $\mathcal{B}_c = \{(\bar{\phi}_{\cdot,c}, \underline{\phi}_{\cdot,c}) \in \{(0,0), (0,1), (1,0), (1,1)\}^{T \cdot F}\}$. Further, the set of all possible conjunctions involving at most D features from $\mathcal{F}$ is

$$\mathcal{T}_c^t(D) = \left\{ (\bar{\phi}, \underline{\phi}) \in \mathcal{B}_c : \sum_{f=1}^F \phi_{f,c}^t \leq D \right\}. \quad (2)$$

When the context allows, we assume $\mathcal{T}(D) = \mathcal{T}_c^t(D)$, for all classes c and conjunctions t , and define $T(D) = |\mathcal{T}(D)|$. We also introduce the zero-one-valued representation of a conjunction:

$$f_{\mathbf{v}_i}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) = \prod_{f: \bar{\phi}_{f,c}^t=1} v_{i,f} \cdot \prod_{f: \underline{\phi}_{f,c}^t=1} (1 - v_{i,f}).$$

Following this definition, and replacing the logical operations AND and OR in the Boolean expressions with linear algebraic counterparts, we define the admissible DNF sets as

$$\tilde{\mathcal{T}}^{\text{CC}}(D) = \left\{ (\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) \in \prod_{c \in \mathcal{C}} \prod_{t \in \mathcal{T}} \mathcal{T}_c^t(D) : \sum_{c \in \mathcal{C}} \max_{t \in \mathcal{T}} f_{\mathbf{v}_i}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) = 1, \forall i \in \mathcal{I} \right\},$$

and $\tilde{\mathcal{T}}^{\text{OC}}(D) = \prod_{c \in \mathcal{C}} \prod_{t \in \mathcal{T}} \mathcal{T}_c^t(D)$. Equivalently, when there is no ambiguity about the CC and OC case, the set of feasible DNFs can be denoted as $\tilde{\mathcal{T}}(D)$.

While Definition 1 describes the general form of a DNF, it is important to emphasize that some clauses are not meaningful for classification: clauses with no literals (empty clauses) and clauses containing both a feature and its negation (contradictory clauses). Although empty and contradictory clauses do not carry semantic meaning in the context of classification, they give rise to distinct mathematical programming reformulations, whose properties can be computationally exploited as investigated in Section 3 below. We formalize these cases in the following definition.

Definition 2 (Empty clauses and contradictory clauses). Let $\phi_{\cdot,c}$ characterize a DNF for some $c \in \mathcal{C}$. We say that the t -th clause of $\phi_{\cdot,c}$ is an empty clause if $\forall f \in \mathcal{F}, \bar{\phi}_{f,c}^t = \underline{\phi}_{f,c}^t = 0$. We say that the t -th clause of $\phi_{\cdot,c}$ is a contradictory clause if $\exists f \in \mathcal{F}, \bar{\phi}_{f,c}^t = \underline{\phi}_{f,c}^t = 1$.

Beyond this distinction, it is important to differentiate between general DNF classifiers and those DNFs that can be induced by a DT structure. Let us consider a full binary tree of depth D , with internal nodes $\mathcal{N}$ and leaves $\mathcal{L}$. For each internal node $n \in \mathcal{N}$, let $b_{n,f} \in \{0,1\}$ indicate whether node n branches on feature $f \in \mathcal{F}$, with $\sum_{f \in \mathcal{F}} b_{n,f} = 1$. For each leaf $\ell \in \mathcal{L}$, let $w_{\ell,c} \in \{0,1\}$ indicate whether leaf ℓ predicts class $c \in \mathcal{C}$, with $\sum_{c \in \mathcal{C}} w_{\ell,c} = 1$, and define the binary indicator $\rho_{n,\ell}$ encoding whether the path from node n to leaf ℓ follows the branch corresponding to $v_f = 1$ for the selected

feature at node n . The class- c region induced by the tree is then

$$h_c^{\text{DT}}(\mathbf{b}, \mathbf{w}; \mathbf{v}) = \sum_{\ell \in \mathcal{L}} w_{\ell, c} r_{\ell}(\mathbf{b}, \mathbf{v}), \quad \text{where} \quad r_{\ell}(\mathbf{b}, \mathbf{v}) = \prod_{n \in \mathcal{P}(\ell)} \left(\sum_{f \in \mathcal{F}} b_{n, f} [\rho_{n, \ell} v_f + (1 - \rho_{n, \ell})(1 - v_f)] \right),$$

where $\mathcal{P}(\ell)$ is the set of internal nodes on the path from the root to leaf ℓ . Since each observation reaches exactly one leaf and each leaf is assigned exactly one class, we have $\sum_{c \in \mathcal{C}} h_c^{\text{DT}}(\mathbf{b}, \mathbf{w}; \mathbf{v}) = 1$, for all $\mathbf{v} \in \{0, 1\}^F$. Assuming that the admissible number of clauses T of a DNF expression is large enough to encode the leaf paths of the tree, we define the set of DT-compatible DNFs of depth D as

$$\tilde{\mathcal{T}}^{\text{DT}}(D) = \left\{ (\bar{\phi}, \underline{\phi}) \in \tilde{\mathcal{T}}^{\text{CC}}(D) : \begin{array}{l} \exists (\mathbf{b}, \mathbf{w}) \text{ satisfying the tree constraints above such that} \\ \max_{t \in \mathcal{T}} f_{\mathbf{v}_i}(\bar{\phi}_{\cdot, c}^t, \underline{\phi}_{\cdot, c}^t) = h_c^{\text{DT}}(\mathbf{b}, \mathbf{w}; \mathbf{v}_i), \quad \forall i \in \mathcal{I}, \forall c \in \mathcal{C} \end{array} \right\}.$$

Hence, $\tilde{\mathcal{T}}^{\text{DT}}(D) \subseteq \tilde{\mathcal{T}}^{\text{CC}}(D) \subseteq \tilde{\mathcal{T}}^{\text{OC}}(D)$. The first inclusion reflects the fact that every DT can be expressed as a DNF (if T is sufficiently large). The inclusion is generally strict: in a DNF formulation, clauses can be selected independently across classes, whereas in a DT formulation all clauses must be generated by a single hierarchical partition of the feature space.

3 Mathematical programming representations

This section investigates exact mathematical programming models for the problem of finding a multi-class Boolean classifier in $\tilde{\mathcal{T}}^j(D)$, where $j \in \{\text{DT}, \text{CC}, \text{OC}\}$, maximizing the number of correctly classified individuals in a training dataset $\{(\mathbf{v}_i, u_i)\}_{i \in \mathcal{I}}$, where the correctness indicator in $\tilde{\mathcal{T}}^j(D)$ is

$$\chi_i^j(\phi) = \begin{cases} \sum_{c \in \mathcal{C}} u_{i, c} \max_{t \in \mathcal{T}} f_{\mathbf{v}_i}(\bar{\phi}_{\cdot, c}^t, \phi_{\cdot, c}^t), & \text{if } j \in \{\text{DT}, \text{CC}\}, \\ \sum_{c \in \mathcal{C}} u_{i, c} \left(\max_{t \in \mathcal{T}} f_{\mathbf{v}_i}(\bar{\phi}_{\cdot, c}^t, \phi_{\cdot, c}^t) \right) \left(\prod_{c' \in \mathcal{C}: c' < c} (1 - \max_{t \in \mathcal{T}} f_{\mathbf{v}_i}(\bar{\phi}_{\cdot, c'}^t, \phi_{\cdot, c'}^t)) \right), & \text{if } j = \text{OC}. \end{cases}$$

In the ordinal case, classes are identified with their pre-specified order. The training-sample accuracy maximization problem can be expressed as

$$\max \sum_{i \in \mathcal{I}} \chi_i^j(\phi), \quad \text{subject to } (\bar{\phi}_{\cdot, \cdot}, \phi_{\cdot, \cdot}) \in \tilde{\mathcal{T}}^j(D). \quad (3)$$

Maximizing classification accuracy under discrete complexity constraints also underlies the approach of Chen and Lee (2018), who develop mixed-integer formulations for linear threshold rules. In our setting, the admissible predictors are multiclass Boolean classifiers, with complexity controlled by

the literal depth and clause budget. The following subsections investigate alternative reformulations of this problem, with a special focus on representation size and equivalence.

3.1 Disjunctive normal forms representation

A binary programming formulation of (3) was introduced by Dash et al. (2018) and Lawless et al. (2023b) for the case of DNF classifiers with $|\mathcal{C}| = 2$, no ordinal structure among classes, and no distinctive assessment for empty and contradictory clauses. A generalized multi-class version covering both CC and OC can be established by introducing three groups of decision variables: $\gamma_{i,t,c}$, representing whether for instance i the t -th clause of class c is true, $\varepsilon_{t,c}$, representing whether the t -th clause of class c is empty, and $\tilde{\gamma}_{i,c}$, representing whether the whole DNF of class c is true for instance i . Based on these variables and introducing the sets Γ^K and Ω_M to identify alternative modeling strategies for handling empty or contradictory clauses, and categorical versus ordinal settings, we have the following reformulation of problem (3):

$$B_M^K(D) = \left\{ \begin{array}{ll} \max & \sum_{i \in \mathcal{I}} \zeta_i \quad (4a) \\ \text{subj. to} & \gamma_{i,t,c} \leq 1 - \left((1 - v_{i,f}) \bar{\phi}_{f,c}^t + v_{i,f} \underline{\phi}_{f,c}^t \right) \quad \forall (i, t, f, c) \quad (4b) \\ & \gamma_{i,t,c} \geq 1 - \sum_{f \in \mathcal{F}} \left((1 - v_{i,f}) \bar{\phi}_{f,c}^t + v_{i,f} \underline{\phi}_{f,c}^t \right) \quad \forall (i, t, c) \quad (4c) \\ & \sum_{f \in \mathcal{F}} \phi_{f,c}^t \leq D \quad \forall (t, c) \quad (4d) \\ & \tilde{\gamma}_{i,c} \leq \sum_{t \in \mathcal{T}} (\gamma_{i,t,c} - \varepsilon_{t,c}) \quad \forall (i, c) \quad (4e) \\ & \tilde{\gamma}_{i,c} \geq \gamma_{i,t,c} - \varepsilon_{t,c} \quad \forall (i, t, c) \quad (4f) \\ & \zeta_i \leq \tilde{\gamma}_{i,u_i} \quad \forall i \quad (4g) \\ & (\zeta, \gamma, \tilde{\gamma}, \phi, \varepsilon) \in \Omega_M^K = \Omega_M \cap \Gamma^K \quad (4h) \\ & \gamma_{i,t,c}, \tilde{\gamma}_{i,c}, \varepsilon_{t,c}, \bar{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t \in \{0, 1\} \quad \forall (i, t, f, c). \quad (4i) \end{array} \right.$$

Here, the indices i , t , f , and c range over the sets $\mathcal{I}$, $\mathcal{T}$, $\mathcal{F}$, and $\mathcal{C}$, respectively. Variable ζ_i indicates whether observation i is correctly classified. Constraints (4b) and (4c) linearize $f_{\mathbf{v}_i}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t)$. Constraint (4d) limits the maximum number of features allowed in each clause. Constraints (4e) and (4f) establish that $\tilde{\gamma}_{i,c}$ is the maximum $\gamma_{i,t,c}$ among all non-empty clauses $t \in \mathcal{T}$. Constraints (4g) indicate that an instance i of class u_i can be considered correctly classified only if the DNF formula corresponding to class u_i evaluates to true. Finally, constraint (4h) specifies the formulation for a given choice of Γ^K and Ω_M , where the index $M \in \{1, \dots, 5\}$ refers to alternative modeling strategies

for empty or contradictory clauses, while $K \in \{\text{CC}, \text{OC}\}$ specifies the classification setting:

$$\Gamma^{OC} := \left\{ \zeta_i \leq 1 - \tilde{\gamma}_{i,c} : i \in \mathcal{I}, c = 1, \dots, u_i - 1 \right\}, \quad \Gamma^{CC} := \left\{ \sum_{c \in \mathcal{C}} \tilde{\gamma}_{i,c} = 1 : i \in \mathcal{I} \right\}. \quad (5)$$

Assuming that classes are ordered by index, Γ^{OC} predicts the label of observation i as the smallest class index whose DNF evaluates to true. Hence, i is correctly classified if and only if the DNF of class u_i is true and all DNFs of classes $c < u_i$ are false. This is enforced by requiring $\zeta_i = 0$ whenever at least one preceding class $c < u_i$ satisfies $\tilde{\gamma}_{i,c} = 1$, i.e., whenever the outcome would be assigned to a class before u_i . Under categorical classification, Γ^{CC} in (5) imposes that exactly one class DNF is true for each i , and that class is the predicted label. Regarding the treatment of empty and contradictory clauses, Ω_M selects a specific combination of the following constraints:

$$\varepsilon_{t,c} = 0, \quad \forall(t, c), \quad (\text{empty clauses not marked}), \quad (6)$$

$$\sum_{f \in \mathcal{F}} \phi_{f,c}^t \geq 1, \quad \forall(t, c), \quad (\text{empty clauses forbidden}), \quad (7)$$

$$\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \leq 1, \quad \forall(t, f, c), \quad (\text{contradictory clauses forbidden}). \quad (8)$$

The sets $\Omega_1, \dots, \Omega_4$ are obtained by choosing which constraints among (6)–(8) are enforced, as summarized in Table 1. Appendix A provides a detailed analysis of these alternatives.

	Empty clauses allowed	Empty clauses forbidden
Contradictory clauses allowed	$\Omega_1 = \{(6)\}$	$\Omega_2 = \{(6), (7)\}$
Contradictory clauses forbidden	$\Omega_3 = \{(6), (8)\}$	$\Omega_4 = \{(6), (7), (8)\}$

Table 1: Definition of the sets Ω_M as a function of whether empty and contradictory clauses are allowed, explicitly indicating the constraints included in each case.

The distinction in Table 1 has important consequences for the characterization of integer solutions. Indeed, if for some class c a clause t is empty (no literal selected), then the clause-evaluation constraints (4b)–(4c) reduce to $\gamma_{i,t,c} = 1$ for all i . Consequently, the disjunction-aggregation constraints (4e)–(4f) force $\tilde{\gamma}_{i,c} = 1$ independently of the remaining clauses, i.e., the class DNF becomes tautologically true and overrides the contribution of all other clauses. For this reason, Ω_2 and Ω_4 forbid empty clauses, while Ω_5 allows them but neutralizes their effect by explicitly identifying them and removing their

contribution from the aggregation. This is formalized as follows:

$$\Omega_5 = \begin{cases} \varepsilon_{t,c} \leq \frac{1}{F} \sum_{f \in \mathcal{F}} (1 - \phi_{f,c}^t) & \forall(t, c) & (9a) \\ \varepsilon_{t,c} \geq \gamma_{i,t,c} - \sum_{f \in \mathcal{F}} \phi_{f,c}^t & \forall(i, t, c) & (9b) \\ \sum_{t \in \mathcal{T}} \varepsilon_{t,c} \leq T - 1 & \forall c & (9c) \\ \varepsilon_{t,c} \in \{0, 1\} & \forall(t, c), & (9d) \end{cases}$$

where (9a)–(9b) set $\varepsilon_{t,c} = 1$ whenever clause t is empty, and (9c) prevents all clauses of a class c from being empty simultaneously. The definition of $\tilde{\gamma}_{i,c}$ is then adjusted so that clauses with $\varepsilon_{t,c} = 1$ are ignored; in particular, if $\gamma_{i,t,c} = 1$ only because clause t is empty, this does not force $\tilde{\gamma}_{i,c} = 1$.

Building on the distinct formulations of sets $\Omega_1, \dots, \Omega_5$ in Table 1, the following three theorems establish equivalence between objective values.

Theorem 1. For $K \in \{\text{CC}, \text{OC}\}$, $B_1^K(D) = B_2^K(D)$.

Theorem 2. For $K \in \{\text{CC}, \text{OC}\}$, $B_5^K(D) = B_1^K(D)$.

Theorem 3. For $K \in \{\text{CC}, \text{OC}\}$, $B_3^K(D) = B_4^K(D)$.

Theorems 1 and 3 show that forbidding empty clauses does not affect the optimal objective value. In contrast, Theorem 2 establishes that neutralizing the effect of empty clauses in the aggregation constraints, corresponding to Ω_5 , is equivalent to allowing empty clauses to remain feasible. Combining these results gives $B_5^K(D) = B_1^K(D) = B_2^K(D) \geq B_3^K(D) = B_4^K(D)$, where the inequality follows from the fact that forbidding contradictory clauses may remove feasible solutions. We next identify a simple condition under which this last inequality also collapses to equality.

Definition 3 (False-satisfiability). *We say that the dataset $\{\mathbf{v}_i\}_{i \in \mathcal{I}}$ is false-satisfiable with D literals if there exists a non-contradictory clause $(\bar{\phi}, \underline{\phi})$ containing no more than D literals such that $f_{\mathbf{v}_i}(\bar{\phi}, \underline{\phi}) = 0$, for all $i \in \mathcal{I}$.*

If the data satisfy the false-satisfiability condition, then $B_5^K(D) = B_1^K(D) = B_2^K(D) = B_3^K(D) = B_4^K(D)$. Indeed, consider an optimal solution of a formulation that allows contradictory clauses, namely Ω_1 or Ω_2 . Each contradictory clause is identically false on the whole feature space and can therefore be replaced, without changing any value of $\gamma_{i,t,c}$ on the training sample, by the non-contradictory all-false clause described in Definition 3. The resulting solution is feasible for the corresponding contradiction-free formulation, Ω_3 or Ω_4 , and has the same objective value.

We now turn our attention to the continuous relaxation of problem (4). The next result shows that, under mild dimensional conditions, the relaxation can attain the trivial upper bound I .

Proposition 1. *Let ζ_i^* , for $i \in \mathcal{I}$, be an optimal solution of the continuous relaxation of problem (4). Under the conditions $\frac{1}{T} \leq 1 - \frac{1}{F}$ and $2 \leq D$, we have $\sum_{i \in \mathcal{I}} \zeta_i^* = I$.*

The weakness identified in Proposition 1 motivates the possibility of treating the binary structure of (4) directly, rather than relying on linear-relaxation-based approaches. This is precisely the role of the QUBO reformulation and its solvability by quantum machines discussed in Section 3.3 below. However, both quantum and binary solvers are themselves highly size-sensitive: in hardware-oriented implementations, logical variables and quadratic couplings must ultimately be accommodated within limited qubit and memory budgets. Thus, as studied in Section 3.3, a QUBO reformulation becomes computationally meaningful only if the underlying DNF encoding is also compressed. We therefore next isolate the dominant source of growth and show how it can be reduced by saturation.

Remark 1. *Problem (4) contains $O(TFC)$ literal-selection and $O(TIC)$ clause-evaluation variables.*

Let us define

$$T_{[\leq D]} = \sum_{k=0}^D \binom{F}{k} 2^k \quad \text{and} \quad T_{[=D]} = \min \left\{ I, \binom{F}{D} 2^D \right\}.$$

If T is chosen to include all distinct clauses using at most D literals over F binary features, then $T = T_{[\leq D]}$. The following theorem establishes that this count can be drastically reduced if shorter clauses are saturated by adding literals on unused features.

Theorem 4. *Any class- c DNF of the form (1) with clauses satisfying $\sum_{f \in \mathcal{F}} \phi_{f,c}^t \leq D$ for all $t \in \mathcal{T}$ is logically equivalent to a DNF in which every non-false clause uses exactly D literals, where $D \leq F$. Contradictory clauses, being identically false, can be removed without altering the DNF.*

Corollary 1. *A saturated DNF formulation can be expressed, without loss of optimality in problem (4), using at most $T = T_{[=D]}$ clause slots per class and imposing $\sum_{f \in \mathcal{F}} \phi_{f,c}^t = D$.*

As a consequence of this saturation result, the literal-selection part of problem (4) can be reduced to $O(T_{[=D]}FC)$ literal-selection variables and $O(T_{[=D]}IC)$ clause-evaluation variables. Moreover, configurations Ω_1 and Ω_2 become pointwise equivalent; similarly, Ω_3 and Ω_4 become pointwise equivalent, since distinctions that rely on the presence of empty clauses disappear. The saturation result also establishes a direct structural connection between DNF- and DT-based formulations. Indeed, any DT of depth D can be embedded into the saturated DNF formulation by expanding the tree into its root-to-leaf paths: each leaf assigned to a class induces a conjunction with at most D

nonredundant literals, which can be replaced, when necessary, by an equivalent disjunction of exact- D clauses using Theorem 4. Conversely, under the exact- D implementation, DT-based solutions form a subset of feasible saturated DNF solutions, and the two representations become comparable within a common clause space.

3.2 Decision tree representation

Bertsimas and Dunn (2017) were the first to propose a mathematical programming formulation of Boolean functions using DTs. This was later refined by Verwer and Zhang (2019), who simplified the constraint structure with the aim of reducing the problem size. More recently, Aghaei et al. (2025) proposed an alternative DT formulation, based on a max-flow design, yielding a stronger continuous relaxation than the original model from Bertsimas and Dunn (2017).³

Following the authors' nomenclature, this section refers to Bertsimas and Dunn (2017)'s formulation as **OCT**, and to Aghaei et al. (2025)'s formulation as **FlowOCT**. To highlight their relationship with the DNF representation and the clause saturation, consider the sets of internal and leaf nodes $\mathcal{N}$ and $\mathcal{L}$, respectively, and recall that $b_{n,f} \in \{0, 1\}$ indicates whether the tree branches on feature $f \in \mathcal{F}$ at node $n \in \mathcal{N}$, and $w_{\ell,c} \in \{0, 1\}$ indicates whether leaf node $\ell \in \mathcal{L}$ predicts class $c \in \mathcal{C}$. In vector form, we write $\mathbf{b} = (b_{n,f})_{n,f}$ and $\mathbf{w} = (w_{\ell,c})_{\ell,c}$. Let $n_{\mathbf{b},i} \in \mathcal{L}$ denote the leaf reached by observation i under the tree structure characterized by $\mathbf{b}$. Since exactly one splitting feature is selected at each internal node and the features are binary, the leaf $n_{\mathbf{b},i}$ is well defined for every observation $i \in \mathcal{I}$. The corresponding classification loss is $1 - \sum_{c \in \mathcal{C}} u_{i,c} w_{n_{\mathbf{b},i},c}$. A compact nonlinear DT formulation, compatible with both **OCT** and **FlowOCT**, can therefore be written as

$$\left\{ \begin{array}{ll} \min & \sum_{i \in \mathcal{I}} \left(1 - \sum_{c \in \mathcal{C}} u_{i,c} w_{n_{\mathbf{b},i},c} \right) \quad (10a) \\ \text{subject to} & \sum_{f \in \mathcal{F}} b_{n,f} = 1 \quad \forall n \in \mathcal{N} \quad (10b) \\ & \sum_{c \in \mathcal{C}} w_{\ell,c} = 1 \quad \forall \ell \in \mathcal{L} \quad (10c) \\ & b_{n,f} \in \{0, 1\} \quad \forall n \in \mathcal{N}, f \in \mathcal{F} \quad (10d) \\ & w_{\ell,c} \in \{0, 1\} \quad \forall \ell \in \mathcal{L}, c \in \mathcal{C}. \quad (10e) \end{array} \right.$$

Here, constraints (10b) select exactly one splitting feature at each internal node, while constraints (10c) assign exactly one predicted class to each leaf. The distinction between the **OCT** and **FlowOCT**

³Another formulation of DTs is provided by Verwer and Zhang (2017), who focus exclusively on linear combinations of continuous features within a tree structure, which therefore departs from Boolean classification.

formulations lies in the way the routing rule defining $n_{\mathbf{b},i}$ is linearized: **OCT** adopts a misclassification-counting approach at each leaf node, whereas **FlowOCT** uses a network-flow formulation that encodes misclassification through flow blockage across the tree (see Appendix C for the detailed MILP formulations of **OCT** and **FlowOCT**).

Remark 2. For a full binary tree with D branching levels, $|\mathcal{N}| = 2^D - 1$ and $|\mathcal{L}| = 2^D$, both **OCT** and **FlowOCT** require $(2^D - 1)F + 2^D C$ binary variables.

Feasible solutions for problem (10) can be easily obtained by applying the greedy algorithm **CART** to the training data. These can be used as warm starts for DNF-based formulations (having often better within-sample errors, due to the relation $\tilde{\mathcal{T}}^{\text{DT}}(D) \subseteq \tilde{\mathcal{T}}^{\text{CC}}(D) \subseteq \tilde{\mathcal{T}}^{\text{OC}}(D)$, highlighted in Section 2), provided that they are converted into the corresponding DNF variables (see Section 4 below for a detailed discussion of a sequential warm-star-based procedure). A DT is naturally converted into a DNF by associating one conjunction with each root-to-leaf path and then taking, for each class, the disjunction of the path clauses associated with leaves assigned to that class.

Definition 4 (DT-to-DNF mapping). Let $(\mathbf{b}, \mathbf{w})$ be a feasible DT. For each leaf $\ell \in \mathcal{L}$, let $P(\ell)$ denote the set of internal nodes on the path from the root to ℓ . For each $n \in P(\ell)$, let $\rho_{n\ell} = 1$ if the path from node n to leaf ℓ follows the branch corresponding to $v_f = 1$ for the feature selected at node n . For each class $c \in \mathcal{C}$, let $\mathcal{L}_c(\mathbf{w}) := \{\ell \in \mathcal{L} : w_{\ell,c} = 1\}$ and choose an injective map $\tau_c : \mathcal{L}_c(\mathbf{w}) \rightarrow \mathcal{T}$. The clause associated with leaf $\ell \in \mathcal{L}_c(\mathbf{w})$ and class c is defined by

$$\begin{aligned} \overline{\phi}_{f,c}^{\tau_c(\ell)} = 1 &\iff \exists n \in P(\ell) : b_{n,f} = 1 \text{ and } \rho_{n\ell} = 1, \\ \underline{\phi}_{f,c}^{\tau_c(\ell)} = 1 &\iff \exists n \in P(\ell) : b_{n,f} = 1 \text{ and } \rho_{n\ell} = 0. \end{aligned}$$

Features that do not appear on the path from the root to ℓ satisfy $\overline{\phi}_{f,c}^{\tau_c(\ell)} = \underline{\phi}_{f,c}^{\tau_c(\ell)} = 0$. The class- c DNF is then obtained as the disjunction of all clauses associated with leaves in $\mathcal{L}_c(\mathbf{w})$.

Definition 4 gives a DNF representation of the same classifier induced by the tree for every class with at least one associated leaf. Indeed, define $g_{i,t,c}(\phi) := f_{v_i}(\overline{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t)$. Then, for every observation i , every class c such that $\mathcal{L}_c(\mathbf{w}) \neq \emptyset$, and every leaf $\ell \in \mathcal{L}_c(\mathbf{w})$, the clause generated by the root-to-leaf path satisfies

$$g_{i,\tau_c(\ell),c}(\phi) = f_{v_i}(\overline{\phi}_{\cdot,c}^{\tau_c(\ell)}, \underline{\phi}_{\cdot,c}^{\tau_c(\ell)}) = \mathbf{1}\{n_{\mathbf{b},i} = \ell\}.$$

Therefore, $\max_{\ell \in \mathcal{L}_c(\mathbf{w})} g_{i,\tau_c(\ell),c}(\phi) = \sum_{\ell \in \mathcal{L}_c(\mathbf{w})} \mathbf{1}\{n_{\mathbf{b},i} = \ell\}$, since exactly one leaf can be reached by observation i . Consequently, the DNF constructed from Definition 4 predicts the same class as the original DT on the training sample. This construction also connects DT with the saturated DNF

representation. In fact, if a root-to-leaf path contains $k < D$ noncontradictory literals, Theorem 4 implies that the associated clause can be replaced by an equivalent disjunction of clauses containing exactly D literals. Concretely, one chooses $D - k$ unused features and appends all possible affirmations and negations of those features. While this DT-to-DNF mapping is immediate when every class that must be represented has at least one corresponding path clause, in practice, a **CART** solution may be sample-degenerate, meaning that one or more classes are never predicted on the training sample.

Definition 5 (Sample-degenerate classifier). *A classifier $U : \{0, 1\}^F \rightarrow \mathcal{C}$ is sample-degenerate with respect to the training sample $\{(\mathbf{v}_i, u_i)\}_{i \in \mathcal{I}}$ if there exists a class $c \in \mathcal{C}$ such that $U(\mathbf{v}_i) \neq c$, for all $i \in \mathcal{I}$. We denote by $\mathcal{C}^0(U) = \{c \in \mathcal{C} : U(\mathbf{v}_i) \neq c, \forall i \in \mathcal{I}\}$ the set of sample-empty classes.*

For a sample-empty class $c^0 \in \mathcal{C}^0(U)$, the corresponding class DNF must be false on every training observation if the DNF representation is to reproduce the DT predictions under categorical classification. Thus, to use a sample-degenerate **CART** solution as a warm-start for a DNF formulation, one must fill the clause slots of each sample-empty class with clauses that are false on the training sample, unless the formulation explicitly neutralizes empty clauses. The required construction depends on Ω_M . If the exact- D saturated implementation is used with $D > 2$, additional literals can be appended to the contradictory clause until the equality cardinality requirement is met; the resulting clause remains identically false.⁴

3.3 QUBO representations

The full binary DNF formulation (4) can be approached by a QUBO representation, since the underlying logical consistency requirements can be enforced by quadratic penalty terms.⁵ This allows eliminating variables $\tilde{\gamma}$ by working directly with the clause-evaluation variables γ and effective clause-activation variables $\alpha_{i,t,c}^M$, whose relation with (γ, ε) is enforced by the following constraints:

$$\mathcal{A}_M = \begin{cases} \left\{ \alpha_{i,t,c}^M = \gamma_{i,t,c}, \quad \forall (i, t, c) \right\}, & M \in \{1, 2, 3, 4\}, \\ \left\{ \begin{array}{ll} \alpha_{i,t,c}^5 \leq \gamma_{i,t,c}, & \forall (i, t, c), \\ \alpha_{i,t,c}^5 \leq 1 - \varepsilon_{t,c}, & \forall (i, t, c), \\ \alpha_{i,t,c}^5 \geq \gamma_{i,t,c} - \varepsilon_{t,c}, & \forall (i, t, c). \end{array} \right\}, & M = 5. \end{cases}$$

⁴Appendix H provides a dedicated procedure that guarantees a non-degenerate decision-tree solution to be used as a warm start in our incremental algorithms.

⁵Among classical exact approaches, equivalent convex mixed-integer quadratic reformulations can be constructed and solved using general-purpose optimization software (Billionnet and Elloumi 2007).

For $M \in \{1, 2, 3, 4\}$, the effective activation coincides with the clause-evaluation variable. Under Ω_5 , the above constraints impose $\alpha_{i,t,c}^5 = \gamma_{i,t,c}(1 - \varepsilon_{t,c})$. Hence, an empty clause satisfies $\gamma_{i,t,c} = 1$ and $\varepsilon_{t,c} = 1$, and therefore contributes $\alpha_{i,t,c}^5 = 0$ to the corresponding class DNF. Exclusivity among class DNFs can then be penalized by

$$\Pi_M^K(\boldsymbol{\alpha}) = \begin{cases} 0 & \text{for ordinal classification,} \\ \sum_{i \in \mathcal{I}} \sum_{\substack{c, c' \in \mathcal{C} \\ c < c'}} \left(\sum_{t \in \mathcal{T}} \alpha_{i,t,c}^M \right) \left(\sum_{t \in \mathcal{T}} \alpha_{i,t,c'}^M \right) & \text{for categorical classification.} \end{cases}$$

For ordinal classification, no such exclusivity penalty is needed, because several class DNFs may evaluate to true and the prediction is determined by the first-true rule. For categorical classification, as an alternative to Γ^{CC} in (5), the penalty term is positive whenever at least one effective clause is active for two distinct classes for the same observation. Once $\tilde{\gamma}$ is removed, the constraints linking correctness to the true class must be written directly in terms of the effective clause-activation terms α^M . We define

$$\hat{\Gamma}_M^{OC} = \left\{ \begin{array}{l} \zeta_i \leq \sum_{t \in \mathcal{T}} \alpha_{i,t,u_i}^M, \quad \forall i, \\ \zeta_i + \alpha_{i,t,c}^M \leq 1, \quad \forall (i, t, c < u_i) \end{array} \right\} \quad \text{and} \quad \hat{\Gamma}_M^{CC} = \left\{ \begin{array}{l} \zeta_i \leq \sum_{t \in \mathcal{T}} \alpha_{i,t,u_i}^M, \quad \forall i, \\ \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \alpha_{i,t,c}^M \geq 1, \quad \forall i \end{array} \right\}.$$

We obtain the following penalized binary quadratic counterpart of (4):

$$\hat{B}_M^K(D) = \left\{ \begin{array}{ll} \max & \sum_{i \in \mathcal{I}} \zeta_i - \rho \Pi_M^K(\boldsymbol{\alpha}) & (11a) \\ \text{s.t.} & (\boldsymbol{\gamma}, \boldsymbol{\phi}, \boldsymbol{\varepsilon}) \in \Omega_M, \quad \boldsymbol{\alpha}^M \in \mathcal{A}_M, \quad (\boldsymbol{\zeta}, \boldsymbol{\alpha}^M) \in \hat{\Gamma}_M^K & (11b) \\ & \text{constraints (4b), (4c), (4d)} & (11c) \\ & \zeta_i, \gamma_{i,t,c}, \varepsilon_{t,c}, \alpha_{i,t,c}^M, \bar{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t \in \{0, 1\} & \forall (i, t, f, c). \quad (11d) \end{array} \right.$$

For categorical classification, the penalty weight $\rho > |\mathcal{I}|$ is sufficiently large to make any violation of class exclusivity dominated by the maximum possible gain in the classification objective, provided that a zero-exclusivity feasible solution exists for the remaining constraints. Under this exact-penalty, optimal solutions of (11) coincide with optimal feasible solutions of the corresponding hard-exclusivity formulation. This said, the clause-evaluation, cardinality, activation, and correctness requirements in problem (11) are still imposed as constraints. To express it as a native QUBO model, we therefore convert (11) into a genuinely unconstrained binary quadratic energy by penalizing all remaining

logical and structural constraints. The inequalities (4b), (4c), and (4d) are first rewritten as equalities using binary-encoded slack variables, while the remaining constraints are collected in Q_M^K . With this notation, the QUBO energy is

$$\mathcal{H}(\mathbf{y}) = - \left(\sum_{i \in \mathcal{I}} \zeta_i - \rho \Pi_M^K(\boldsymbol{\alpha}) \right) + Q_M^K(\mathbf{y}) + \theta_1 W_1(\mathbf{y}) + \theta_2 W_2(\mathbf{y}) + \theta_D W_D(\mathbf{y}) = \mathbf{y}^\top J \mathbf{y} + \text{const.},$$

where $\mathbf{y}$ contains the original binary variables⁶ $\mathbf{x}$ of problem (11), together with the ancillary binary variables required to encode the slack variables. The matrix J is obtained by collecting the quadratic terms in the objective and in the penalties below, up to additive constants. The weights $\theta_1, \theta_2, \theta_D$ and the weights entering Q_M^K are chosen sufficiently large to enforce the corresponding constraints at optimality (Lucas 2014).⁷ The penalties associated with (4b), (4c), and (4d) are

$$W_1(\mathbf{y}) = \sum_{i \in \mathcal{I}} \sum_{c \in \mathcal{C}} \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \left(\gamma_{i,t,c} + m_{i,t,c,f} + z_{i,t,c,f}^{(1)} - 1 \right)^2, \quad (12)$$

$$W_2(\mathbf{y}) = \sum_{i \in \mathcal{I}} \sum_{c \in \mathcal{C}} \sum_{t \in \mathcal{T}} \left(\gamma_{i,t,c} + \sum_{f \in \mathcal{F}} m_{i,t,c,f} - 1 - s_{i,t,c}^{(2)} \right)^2, \quad (13)$$

$$W_D(\mathbf{y}) = \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \left(\sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right) + s_{t,c}^{(D)} - D \right)^2. \quad (14)$$

Here, $z_{i,t,c,f}^{(1)}$ is binary, while $s_{i,t,c}^{(2)}$ and $s_{t,c}^{(D)}$ are nonnegative integer slack variables encoded in binary, and $m_{i,t,c,f} := (1 - v_{i,f}) \bar{\phi}_{f,c}^t + v_{i,f} \underline{\phi}_{f,c}^t$. The signs of the slack variables reflect the direction of the original inequalities: W_1 encodes $\gamma_{i,t,c} + m_{i,t,c,f} \leq 1$, W_2 encodes $\gamma_{i,t,c} + \sum_f m_{i,t,c,f} \geq 1$, and W_D encodes $\sum_f (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) \leq D$. The slack variables $s_{i,t,c}^{(2)}$ and $s_{t,c}^{(D)}$ can be encoded using enough binary digits to represent the values $0, \dots, D$. The remaining penalty Q_M^K collects the correctness constraints, the activation-linking constraints in $\mathcal{A}_M$, and the formulation-specific constraints defining Ω_M . The activation-linking component can be taken as

$$Q_{\mathcal{A},M} = \begin{cases} \lambda^{\mathcal{A}} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} (\alpha_{i,t,c}^M - \gamma_{i,t,c})^2, & M \in \{1, 2, 3, 4\}, \\ \lambda^{\mathcal{A}} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} (\gamma_{i,t,c} - \gamma_{i,t,c} \varepsilon_{t,c} - 2\gamma_{i,t,c} \alpha_{i,t,c}^5 + 2\varepsilon_{t,c} \alpha_{i,t,c}^5 + \alpha_{i,t,c}^5), & M = 5. \end{cases}$$

⁶The original binary variables of problem (11) are $\zeta_i, \gamma_{i,t,c}, \varepsilon_{t,c}$ when retained, $\alpha_{i,t,c}^M, \bar{\phi}_{f,c}^t$ and $\underline{\phi}_{f,c}^t$, which collectively form the vector $\mathbf{x}$.

⁷A sufficient conservative choice is to make each such weight larger than an upper bound on the largest possible improvement in the non-penalized part of the energy that could be obtained by violating the associated constraint. In the categorical case, such a bound must account for both the classification term and the variation of the exclusivity penalty, for instance through a bound depending on $|\mathcal{I}| + \rho |\mathcal{I}| \binom{C}{2} T^2$.

For correctness, we use

$$\begin{cases} Q_{\text{class}}^{CC} = \lambda^{CC,1} \sum_{i \in \mathcal{I}} \left(\sum_{t \in \mathcal{T}} \alpha_{i,t,u_i}^M - \zeta_i - s_i^{CC} \right)^2 + \lambda^{CC,2} \sum_{i \in \mathcal{I}} \left(\sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \alpha_{i,t,c}^M - 1 - s_i^{CC,0} \right)^2, \\ Q_{\text{class}}^{OC} = \lambda^{OC,1} \sum_{i \in \mathcal{I}} \left(\sum_{t \in \mathcal{T}} \alpha_{i,t,u_i}^M - \zeta_i - s_i^{OC} \right)^2 + \lambda^{OC,2} \sum_{i \in \mathcal{I}} \sum_{\substack{c \in \mathcal{C} \\ c < u_i}} \sum_{t \in \mathcal{T}} \zeta_i \alpha_{i,t,c}^M. \end{cases}$$

The slack variables s_i^{CC} , $s_i^{CC,0}$ and s_i^{OC} are nonnegative integer variables encoded in binary. In particular, s_i^{CC} and s_i^{OC} need to represent values up to T , while $s_i^{CC,0}$ needs to represent values up to $CT - 1$. The first term in each expression enforces that ζ_i can be one only if the DNF of the true class is activated by at least one effective clause. The second term in Q_{class}^{CC} enforces activation of at least one class DNF, whereas categorical exclusivity is enforced by Π_M^K . The second term in Q_{class}^{OC} enforces the first-true ordinal logic. The formulation-specific components of Q_M^K are provided in Appendix A; if ε is retained for $M \in \{1, 2, 3, 4\}$, the fixing $\varepsilon_{t,c} = 0$ is enforced by adding $\lambda^\varepsilon \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \varepsilon_{t,c}$ to Q_M^K , while for $M = 5$ the corresponding terms are obtained by penalizing the defining constraints of Ω_5 . The saturation result in Theorem 4 and its finite-sample implication have a direct implication for the QUBO size. Under the saturated non-contradictory implementation, the clause index satisfies $t \in \{1, \dots, T\}$, where $T = T_{[=D]}$, and the cardinality constraint is imposed as $\sum_{f \in \mathcal{F}} \phi_{f,c}^t = D$, for all $t \in \{1, \dots, T\}$ and $c \in \mathcal{C}$. Therefore, in the slot-based encoding used in the variable count below, non-contradiction must be imposed explicitly, for instance through the quadratic penalty $Q_{\text{nc}} = \lambda^{\text{nc}} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \sum_{f \in \mathcal{F}} \bar{\phi}_{f,c}^t \phi_{f,c}^t$. The cardinality penalty no longer requires a slack-variable encoding, because it becomes the equality penalty $\left(\sum_{f \in \mathcal{F}} \phi_{f,c}^t - D \right)^2$. Empty clauses are ruled out by construction. Consequently, the variables $\varepsilon_{t,c}$ and the penalties associated with empty-clause detection or elimination are no longer needed in the saturated implementation. In particular, the distinction between Ω_1 and Ω_2 disappears, as does the distinction between Ω_3 and Ω_4 ; if non-contradiction is imposed globally and empty clauses are ruled out by exact saturation, all formulations $\Omega_1, \dots, \Omega_5$ collapse to the same pointwise representation on the nonempty, non-contradictory clause space. This implementation is without loss whenever false or unused class clauses can be represented, for instance under the false-satisfiability condition introduced above. In this saturated non-contradictory implementation, $\alpha_{i,t,c}^M$ can be identified with $\gamma_{i,t,c}$ and does not need to be introduced as an additional variable. Before adding the classification-specific slack variables, the saturated QUBO contains

$$N_{\text{QUBO}}^{\text{sat}}(D) = I + I \cdot T_{[=D]} \cdot C + 2F \cdot T_{[=D]} \cdot C + I \cdot T_{[=D]} \cdot C \cdot F + I \cdot T_{[=D]} \cdot C \lceil \log_2(D + 1) \rceil$$

binary variables.⁸ This expression shows that the saturation principle has a direct practical implication for quantum implementation. By replacing the pre-saturation clause count $T_{[\leq D]}$ with $T_{[=D]}$, and by eliminating the ancillary variables associated with empty-clause handling and with the slack encoding of $\sum_f \phi_{f,c}^t \leq D$, saturation substantially reduces the number of binary variables that must be embedded into a QUBO instance.

It is worth noticing that this QUBO representation constitutes the native input of quantum-annealing solvers, providing a direct way to explore the combinatorial landscape induced by the introduced penalties without relying on convex relaxations. In our setting, the relevance of quantum annealing is tied to the saturation-based reduction: by shrinking the clause space and removing unnecessary ancillary variables, the formulation becomes more compatible with the limited qubit budgets and connectivity constraints of current devices. This is particularly useful in regimes where the nonconvex quadratic term and the weakness of continuous relaxations limit the effectiveness of standard branch-and-cut approaches. Appendix D provides background and implementation details on quantum annealing in our context.

4 Incremental learning

Throughout this section, we introduce the hyperparameter $D_{\max} < F$ as the maximum literal-depth allowed in a Boolean expression. In the DT representation, this parameter coincides with the maximum tree depth. In the DNF representation, it controls the maximum number of selected literals that may appear in each clause.

To formalize this dependence, recall that $\mathcal{T}_c^t(D)$ has been defined in (2) as the set of all possible conjunctions involving at most $D \leq D_{\max}$ selected literals over $\mathcal{F}$. The size of problems (4) and (10) grows rapidly with D (see Remarks 1 and 2). The computational importance of exploiting preliminary information in mixed-integer classification is also illustrated by Chen and Lee (2018), whose warm-start strategy uses preliminary choice-probability estimates to tighten the coefficient domain of their binary prediction model. Here, we exploit the structure of Boolean representations to transfer feasible classifiers between successive optimization problems. The incremental strategies below mitigate the growth in problem size by solving a sequence of progressively richer problems, while carrying forward and, when necessary, lifting high-quality incumbent solutions. The practical motivation is that accurate Boolean classifiers often use only a small fraction of the worst-case clause

⁸For categorical classification, the constraints $\zeta_i \leq \sum_{t=1}^T \gamma_{i,t,u_i}$ and $\sum_{c \in \mathcal{C}} \sum_{t=1}^T \gamma_{i,t,c} \geq 1$ require, respectively, at most $\lceil \log_2(T+1) \rceil$ and $\lceil \log_2(CT+1) \rceil$ ancillary bits per observation. For ordinal classification, the constraint $\zeta_i \leq \sum_{t=1}^T \gamma_{i,t,u_i}$ requires at most $\lceil \log_2(T+1) \rceil$ ancillary bits per observation, whereas the first-true restrictions can be enforced by quadratic penalties of the form $\zeta_i \gamma_{i,t,c}$ for $c < u_i$, without additional slack variables in the saturated case.

space. Figure 1 illustrates this point by comparing the number of clauses induced by the **CART** algorithm solutions with the pre- and post-saturation reference counts (in logarithmic scale).

We consider two incremental strategies. The first is a *single incremental* strategy, in which only D is increased. This is natural for DT formulations, because the number of leaves is determined by the tree depth. It is also natural for exact- D saturated DNF formulations when the clause budget is fixed as a function of D . The second is a *double incremental* strategy, in which both D and the DNF clause budget T are increased. This second strategy exploits a distinctive feature of DNF formulations: the number of clause slots can be controlled independently of the maximum number of literals, provided that one works with a truncated dictionary. Both single and double incremental procedures benefit from efficiently computable warm-start solutions, such as those obtained by the **CART** algorithm or, for QUBO-based DNF encodings, by quantum-annealing solvers applied to the QUBO representation (11), as clarified below.

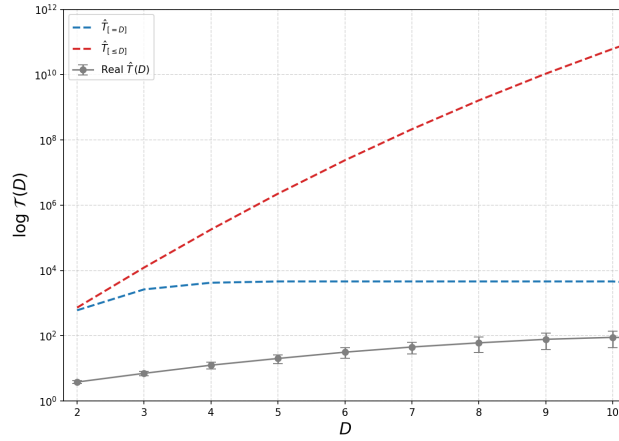


Figure 1: Number of clauses embedded in the **CART** solutions, as a function of the depth D , evaluated on real-world machine learning classification datasets. The grey curve reports the average number of clauses across the considered benchmarks, with error bars showing the across-dataset variation. The two dashed lines mark reference counts: the complete pre-saturation count $T_{[≤D]}$ (red) and the saturation count $T_{[=D]}$ (blue).

Single incremental procedure. The single incremental strategy applies when the model size, or the selected clause budget, is determined once the literal-depth parameter D is fixed. This is the case for DT formulations, where a complete binary tree of depth D has 2^D leaves. It is also the natural strategy for exact- D saturated DNF formulations, because the saturated clause budget is determined by D and set to $T_{[=D]}$. In this setting, the role of saturation is twofold. First, it replaces the larger non-saturated dictionary of clauses with at most D literals by the reduced exact- D dictionary. Second, it determines how incumbent DNF solutions can be transferred from one value of D to the next. Indeed, a clause with exactly D literals cannot be directly reused in a formulation requiring exactly $D + 1$ literals. It must first be expanded into clauses of size $D + 1$. The following

lemma formalizes this lifting step.

Lemma 1. *Given a DNF $(\bar{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t)$ with T clauses per class, in which every clause has exactly D literals and $D < F$, it can be equivalently represented as a DNF $(\bar{\phi}_{f,c}'^t, \underline{\phi}_{f,c}'^t)$ with $2T$ clauses per class, in which every clause has exactly $D + 1$ literals.*

Lemma 1 shows that preserving the same Boolean function while moving from exact depth D to exact depth $D + 1$ may require increasing the number of clauses. In particular, starting from a two-clause one-literal partition, repeated application of this operation gives the schedule $T(D) = 2^D$. The full exact- D saturated formulation instead uses the predetermined clause budget $T_{[=D]}$. The single incremental procedure iterates over D and can be summarized as follows:

- (i) for each value of D , compute a heuristic solution and use it as an initial incumbent;
- (ii) solve the selected DT or exact- D DNF formulation at the current value of D ;
- (iii) keep the best feasible incumbent found so far and use it to initialize the next iteration;
- (iv) in exact- D DNF formulations, lift the previous incumbent to depth $D + 1$ using Lemma 1, whenever the selected exact- $(D + 1)$ clause budget contains enough slots to accommodate the lifted representation.

The full pseudo-code of the single-incremental learning is reported in Appendix I, Algorithm 2.

Double incremental procedure. The double incremental strategy is specific to DNF-based formulations, but it must be distinguished from the exact- D saturated implementation discussed above. Under the saturation principle, once the equality constraint $\sum_{f \in \mathcal{F}} \phi_{f,c}^t = D$ is imposed and the full exact- D saturated representation is used, the admissible clause budget is fixed by D and set to $T_{[=D]}$. In that case, T is no longer an independent modeling parameter: increasing D automatically determines the corresponding saturated budget. Hence, the exact- D saturated implementation naturally falls under the single incremental logic. The double incremental procedure is therefore intended for the non-saturated DNF formulation, where clauses are allowed to contain at most D literals, $\sum_{f \in \mathcal{F}} \phi_{f,c}^t \leq D$, and where the number of clause slots T can be chosen independently of D . In this setting, for a fixed value of D , the reference full dictionary contains $T_{[\leq D]}$ possible clauses. Solving the full formulation with all these clauses is usually computationally prohibitive. The purpose of the double incremental strategy is therefore to replace this single large problem by a sequence of smaller DNF formulations with progressively increasing clause budgets.

For each value of D , we consider a heuristic solution (such as **CART**), which provides a feasible, possibly repaired, warm start and an estimate of the number of clauses needed to encode a tree-based classifier in DNF form. We denote this number by $T_{\text{WS}}(D)$. Since the double incremental procedure is used in

the non-saturated formulation, the root-to-leaf path clauses obtained from this heuristic do not need to be expanded to exact size D ; they already satisfy the inequality cardinality restriction. The inner loop starts from $T_{\min}(D) = \max\{2D, T_{\text{WS}}(D)\}$, and increases the clause budget up to a prescribed upper bound $T_{\max}(D)$, with $T_{\max}(D) \leq T_{[\leq D]}$. In practice, $T_{\max}(D)$ is chosen as a computational cap whenever this value is substantially smaller than the full pre-saturation dictionary. The double incremental procedure boils down to the following sequence of steps for each D :

- (i) compute a heuristic solution and convert it into a DNF warm start using Definition 4;
- (ii) compute $T_{\text{WS}}(D)$ from this mapped solution and solve a sequence of non-saturated DNF formulations with clause budgets $T = T_{\min}(D), \dots, T_{\max}(D)$;
- (iii) at each value of T , warm start the formulation with the best available feasible solution, coming either from the heuristic solution obtained in (i), from the previous solution of problem (4) at a smaller value of T , or from the best solution obtained at a smaller value of D , embedded in the current (D, T) formulation.

This method retains the best incumbent found across all tested values of D and T . For fixed D , the exact optimal value of the non-saturated problem is nondecreasing in T , because any solution using fewer clauses can be embedded into a formulation with more clause slots by duplicating existing clauses, based on the formulation Ω_M . The full pseudo-code of the double incremental procedure is reported in Appendix I, Algorithm 3.

5 Numerical experiments on real data

This section evaluates the modeling reductions and warm-start strategies developed in the paper on benchmark Boolean classification instances. The computational experiments follow the three methodological components introduced above. First, we compare the DNF and DT formulations of Section 3. Second, we assess the incremental procedures of Section 4, with special attention to the role of incumbent reuse across increasing values of the depth parameter. Third, we test the QUBO-based quantum annealing approach of Section 3.3 as a heuristic incumbent generator for the DNF MILPs. In all experiments, objective values are reported in the natural units of the optimization models, namely the number of correctly classified training observations.

5.1 Benchmark datasets and experimental setup

We use seven publicly available datasets from the UCI Machine Learning Repository (Kelly et al. 2023). These datasets provide standard Boolean classification benchmarks with different numbers of observations, binary-encoded features, and classes, as reported in Table 2.

Data Name	I	$I^\dagger$	$I^\ddagger$	F	$F^\dagger$	$F^\ddagger$	C
monk1	554	124	124	11	11	15	2
monk2	599	169	169	11	11	15	2
monk3	552	122	122	11	11	15	2
bln-scale	625	625	625	16	4	20	3
tic-tac-toe	958	958	958	18	18	18	2
car-eval	1728	1728	1728	15	15	20	4
kr-kp	3196	-	3196	37	-	38	2

Table 2: Dataset sizes used in our experiments. I and F denote the number of instances and binary-encoded features, respectively. $I^\dagger$ and $F^\dagger$ correspond to the values reported in Bertsimas and Dunn (2017); the **kr-kp** dataset was not considered in that study. $I^\ddagger$ and $F^\ddagger$ denote the values reported in Aghaei et al. (2025). C denotes the number of classes, which is common across all three studies. Differences in the **monk** sample sizes arise because we use the full datasets, whereas Bertsimas and Dunn (2017) and Aghaei et al. (2025) rely on subsets. Differences in the number of features are due to alternative preprocessing choices, detailed in Appendix E.

The same instances, or closely related versions, have been used by Bertsimas and Dunn (2017) and Aghaei et al. (2025) in computational studies of exact DT formulations. Table 2 reports the dimensions used in our experiments and compares them with those reported in these references. Unlike Bertsimas and Dunn (2017) and Aghaei et al. (2025), who use reduced samples in some cases, we retain the full datasets. This choice is consistent with the purpose of this section: we do not study out-of-sample validation, but the scalability of the proposed equivalence-based reductions and incremental optimization. Appendix E provides the details of feature encoding and preprocessing of the seven datasets reported in Table 2.

For these instances, we compare six computational approaches: (1) the IBM ILOG CPLEX implementation of the **FlowOCT** formulation described in Appendix C, following the exact DT line of Bertsimas and Dunn (2017) and Aghaei et al. (2025); (2) the IBM ILOG CPLEX implementation of the single-incremental warm-started version of **FlowOCT**, based on Algorithm 2; (3) the IBM ILOG CPLEX implementation of the DNF formulations of Subsection 3.1, in the spirit of the DNF optimization approach of Dash et al. (2018), solved either directly or with a **CART**-based warm start; (4) the IBM ILOG CPLEX implementation of the double-incremental warm-started DNF approach of Algorithm 3; (5) the IBM ILOG CPLEX implementation of the single-incremental DNF scheme induced by Algorithm 2, where the clause budget is tied to the depth; (6) the hybrid quantum annealing approach applied to the QUBO reformulations of the DNF models in Section 3.3, consistently with the quantum adiabatic and annealing perspective of Kadowaki and Nishimori (1998) and Albash and Lidar (2018).

For methods (1)–(5), all routines were implemented in **C++** using the IBM ILOG CPLEX Callable Library. Experiments were run on a workstation equipped with a 13th Gen Intel(R) Core(TM) i9-13900K processor and 128 GB of RAM, under Windows 11 Professional, using eight CPLEX

threads.⁹ The quantum annealing experiments were conducted on the D-Wave Advantage 6.4 system, with 5,616 qubits and 40,088 couplers. Access to both hybrid and pure quantum solvers was obtained through D-Wave’s Ocean software development kit, which supports QUBO formulation, minor embedding, and execution on the quantum processing unit.

5.2 Numerical tests

We now compare the formulations of Sections 3 and 3.3, together with the solution procedures proposed in Section 4. The non-incremental benchmark solves the DNF or DT formulation for a single fixed value of the depth parameter D . For each dataset, this value is chosen as the largest depth reached by the corresponding incremental procedures, so that the comparison isolates the effect of the algorithmic strategy rather than that of a different depth budget.

At each iteration of an incremental procedure, the warm start is selected as the best available incumbent among the inherited solution, the previous clause-budget solution when applicable, and the **CART** solution at the current depth. In the double-incremental DNF setting of Algorithm 3, several values of the clause budget T are explored at each depth. In our implementation, for $D \leq 4$ we consider only $T = 2^D$, whereas for $D > 4$ we increase T from $2D + 1$ up to 2^D in steps of size D . This design exploits a feature specific to DNF models: unlike DTs, their clause budget can be controlled independently of the depth.

We first focus on the non-saturated DNF formulation (4), where the incremental logic is richest because the clause budget is a genuine modeling parameter. Competing methods are ranked lexicographically: a method is preferred if it attains a larger objective value, and, in case of ties, if it does so in a smaller total running time. Figure 2 aggregates these comparisons over the dataset-configuration pairs for which the **CART** starting solution does not already close the optimization problem. If **CART** attains the maximum possible objective value, further refinement is unnecessary for the in-sample objective. Two messages emerge from Figure 2. First, the most competitive MILP strategies are those that exploit strong incumbents, either through a direct **CART**-based warm start or through incremental reuse across depths. This supports the paper’s warm-start-centric viewpoint: for moderate instances, a good initial classifier may be sufficient, whereas for harder instances the progressive lifting of incumbents becomes increasingly useful. Second, the ranking across the Ω_M formulations is mainly computational rather than structural. The equivalence results of Subsection 3.1 concern exact optimal objective values. By contrast, Figure 2 is based on time-limited runs with positive MIP gaps. Hence, formulations that are equivalent at the exact optimum may still behave differently computationally.

⁹In contrast, Aghaei et al. (2025) relied on a single-core SL250s Xeon CPU with 4 GB of RAM, which limited the size of the instances they could solve; in particular, the **kr-kp** dataset was only solved up to $D_{\max} = 3$.

In the present experiments, Ω_3 is the most competitive specification, whereas Ω_5 is usually dominated. This is consistent with the fact that Ω_5 introduces additional variables and constraints to neutralize empty clauses, which may enlarge the search space without improving the target optimal value.

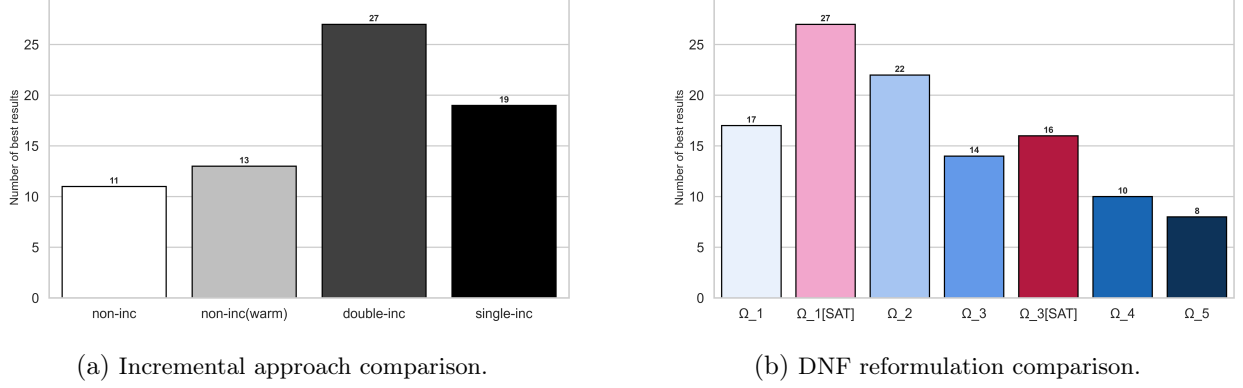


Figure 2: Aggregated counts across all CC and OC configurations. The plots include only the instances for which the **CART** starting solution does not already close the optimization problem. Panel 2a compares the classical solution strategies: non-incremental, non-incremental with warm start, single incremental, and double incremental. Panel 2b compares the DNF reformulations, that is, the Ω_M configurations.

We next compare the DT formulation (10) with the DNF formulation (4). On the DT side, we use the max-flow formulation of Aghaei et al. (2025), which is the strongest DT benchmark considered in this paper. Since the number of leaves in a DT is tied to the depth, the double-incremental strategy of Algorithm 3 has no direct **FlowOCT** counterpart. The natural DT benchmark is therefore the single-incremental procedure of Algorithm 2, optionally combined with a **CART**-based warm start. Table 3 reports the three **FlowOCT** variants, while Table 4 reports the DNF variants.

	SI FlowOCT		NI FlowOCT (without WS)		NI FlowOCT (with WS)	
	Obj(Time)	D^*	Obj(Time)	D^*	Obj(Time)	D^*
monk1	554(12.80)	5	554(4.74)	5	554(3.48)	5
monk2	—	—	—	—	—	—
monk3	546(124.40)	6	546(80.21)	6	546(330.03)	6
bln-scale	462(38.12)	4	473(173.70)	4	464(159.95)	4
tic	954(260.60)	8	626(330.04)	8	954(330.01)	8
car-eval	1686(261.78)	8	1172(330.02)	8	1686(330.02)	8
krkp	3007(96.48)	5	2159(330.02)	5	3007(330.02)	5

Table 3: **FlowOCT** results across datasets. For each configuration, we report the objective value, computational time in minutes, and selected depth D^* . From left to right, the columns denote the single-incremental method (SI), the non-incremental method (NI), and the non-incremental method with **CART**-based warm start (NI-WS). The results for **monk2** are not reported because **CART** returned a perfectly classified solution.

The comparison confirms the value of the DNF representation. The best DNF result matches or improves upon the best **FlowOCT** objective on every dataset for which both methods are reported. The gains are especially visible on **balance-scale**, **tic-tac-toe**, **car-evaluation**, and **kr-kp**. On

the **monk** instances, objective values often tie, but the DNF methods typically reach these values with smaller computational effort.

	SI		DI		NI(wo-WS)		NI(with WS)	
	Obj(Time)	(D^*, T^*)	Obj(Time)	(D^*, T^*)	Obj(Time)	(D^*, T^*)	Obj(Time)	(D^*, T^*)
monk1	554 (4.58)	(3,8)	554 (2.22)	(3,9)	554 (144.39)	(4,554)	554 (63.79)	(3,554)
monk2	599 (65.09)	(6,64)	599 (58.02)	(6,31)	597 (330.43)	(6,599)	599 (35.20)	(6,599)
monk3	546 (5.25)	(5,32)	546 (5.18)	(5,16)	546 (360.42)	(6,552)	546 (330.05)	(6,552)
bln-scale	579 (259.30)	(8,256)	578 (495.79)	(7,43)	549 (325.99)	(8,624)	583 (328.94)	(8,624)
tic	955 (271.03)	(8,256)	958 (392.50)	(8,49)	0 (27.46)	(8,958)	958 (65.25)	(8,958)
car-eval	1686 (259.12)	(8,256)	1696 (602.26)	(8,41)	-1 (48.87)	(8,256)	-1 (48.87)	(8,256)
krkp	3008 (100.68)	(5,32)	3007 (140.55)	(5,16)	-1 (330.09)	(5,32)	-1 (330.09)	(5,32)

Table 4: DNF results across datasets. From left to right, the columns correspond to the single-incremental method (SI), double-incremental method (DI), non-incremental method without warm start (NI), and non-incremental method with **CART**-based warm start (NI-WS). For each method, we report the best result over the tested Ω_M configurations, the objective value with computational time in minutes, and the selected parameters (D^*, T^*) . A value of -1 indicates that no feasible solution was found within the time limit.

Although this pattern is consistent with the links between DT and DNF representations, the numerical comparison is performed under the algorithmic schemes of Section 4. In that setting, DNF formulations benefit from an additional degree of freedom through the clause budget T , whereas DT formulations tie the number of terminal regions to the depth. Tables 3 and 4 show that this flexibility is computationally useful.

Figure 3 summarizes this comparison in normalized objective-time space, reinforcing the evidence that single-incremental **FlowOCT** improves the non-incremental benchmark, though the best DNF solutions are usually located closer to the region of high objective value and low computation time. Table 5 quantifies this comparison through method centroids in the normalized objective-time space. The **Best** DNF envelope attains the highest normalized objective (1.000) and one of the lowest normalized times (0.324), placing it close to the ideal upper-left corner and ahead of all **FlowOCT** variants. Relative to NI **FlowOCT**, it improves the normalized objective by 18.8% while being 35.1% faster; relative to NI-WS **FlowOCT**, it improves the objective by 3.6% while being 46.7% faster. Among DT methods, SI **FlowOCT** is the strongest, with normalized objective 0.964 and time 0.317. Relative to this benchmark, SI DNF improves the objective (0.998 versus 0.964) at essentially the same running time, while DI DNF reaches a comparable objective (0.999) at higher cost (0.519). This trade-off is expected, since DI DNF is not tied to the tree-like clause schedule 2^D discussed after Lemma 1 and can exploit a richer non-saturated clause dictionary.

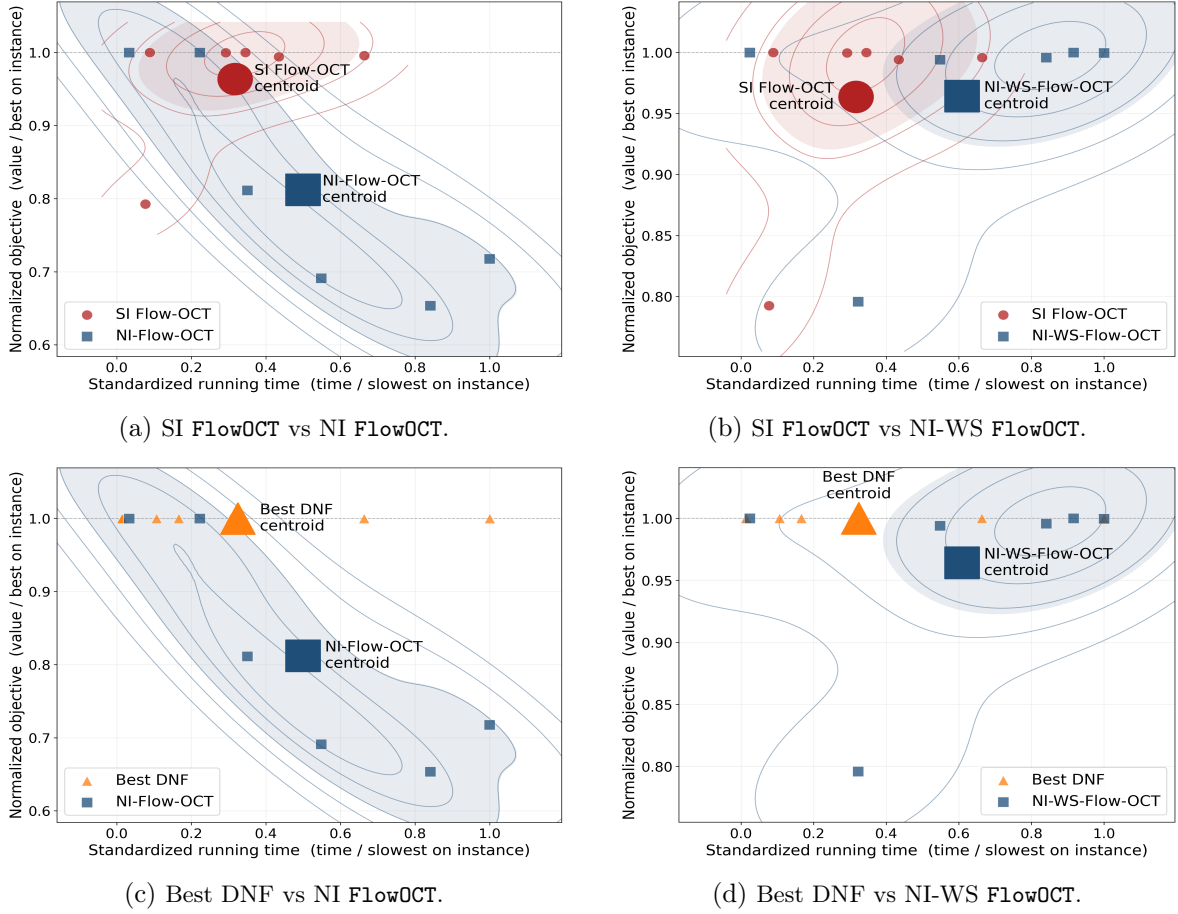


Figure 3: Two-dimensional comparison of methods in normalized quality–cost space. The horizontal axis is the standardized running time (instance time divided by the slowest method on that instance, so lower is faster) and the vertical axis is the normalized objective (instance value divided by the best objective across methods, so 1.0 is best). The large marker of matching color is that method’s centroid (the mean over the instances it solved). In panels (a) and (b), red circles are SI FlowOCT and blue squares the NI FlowOCT variant. In panels (c) and (d), orange triangles are the best DNF-based approach and blue squares the NI FlowOCT variant.

We finally compare two incumbent generators for the DNF MILPs: **CART** and the QUBO-based quantum annealing routine of Section 3.3. Both aim to produce a high-quality DNF assignment that can be reused as a warm start in downstream exact MILP solves. **CART** is run independently at each depth and then converted into DNF form. The quantum annealing routine is instead embedded in a single-incremental scheme, denoted by **SI-QA**, in which the QUBO at depth D is warm-started from the incumbent obtained at depth $D - 1$.

The QUBO formulation exploits the saturation result of Theorem 4: in the saturated implementation, forcing every clause to contain exactly D literals avoids the slack-variable machinery that would otherwise be needed to encode the inequality cardinality restriction. Even so, current annealers cannot encode the largest datasets in full at every depth. The comparison below therefore uses a subsampling protocol: for each dataset we draw six random 300-row subsamples with **Python** seeds 40, 42, 44, 46, 48, 50, and run both **SI-QA** and **CART** on the same subsamples.

Method	Normalized centroid		ΔObj vs. Best DNF (%)
	Obj_{norm}	$\text{Time}_{\text{norm}}$	
SI FlowOCT	0.964	0.317	−3.6
NI FlowOCT	0.812	0.499	−18.8
NI-WS FlowOCT	0.964	0.608	−3.6
SI	0.998	0.313	−0.2
DI	0.999	0.519	−0.1
NI(wo-WS)	0.985	0.914	−1.5
NI(with WS)	1.000	0.459	0.0
Best DNF	1.000	0.324	—

Table 5: Centroids of each method in the normalized quality–cost space, averaged over the instances each method solved. Obj_{norm} is the per-instance objective divided by the best objective across methods (higher is better, 1.0 = best). $\text{Time}_{\text{norm}}$ is the per-instance running time divided by the slowest across methods (lower is better, 1.0 = slowest). The last column reports the percentage gap in normalized objective relative to *Best DNF* (negative = lower objective than Best DNF). *Best DNF* is the per-instance upper envelope of the four DNF variants.

Disaggregated results are in Appendix G. Figure 4 shows that **SI-QA** is a competitive incumbent generator for the DNF MILP. Panel 4a places it above **CART** in median and upper tail, and the paired comparison confirms this: the mean **SI-QA**–**CART** gap is 18.8 correctly classified instances under categorical classification (Cohen’s $d_z = 0.96$, paired $t = 6.34$, $p < 10^{-7}$; Wilcoxon $p < 10^{-7}$) and 61.4 instances under ordinal classification (Cohen’s $d_z = 3.47$, paired $t = 23.04$, $p < 10^{-25}$; Wilcoxon $p < 10^{-8}$). Unlike **CART**, which must be translated into DNF clauses and may require repair when the induced tree is sample-degenerate, **SI-QA** works directly with the QUBO variables; see Section 3.2, Definitions 4 and 5, and Appendix H. Panel 4b compares the two Ω equivalence classes within **SI-QA**. Their distributions are nearly indistinguishable, in line with the equivalence results of Section 3.1, according to which $B_5^K(D) = B_1^K(D) = B_2^K(D) \geq B_3^K(D) = B_4^K(D)$ at the exact optimum. Panel 4c contrasts the two classification modes: **OC** dominates **CC** both in median and in the upper tail (mean **CC**–**OC** gap $\Delta = -42.6$ correctly classified instances, $d_z = -1.72$, paired $t = -11.40$, $p < 10^{-13}$). This is consistent with the fact that **OC** induces a less constrained QUBO, while still producing valid DNF classifiers. Figure 8 provides the corresponding dataset-level view. Table G.2 reinforces these conclusions: whenever annealing is feasible, **SI-QA** matches or improves on **CART** at the same final depth D^* , often by a large margin, while keeping the same model complexity. These gains should be read together with the current limitations of quantum annealing. Present-day machines remain size-constrained and sensitive to embedding quality and hardware noise. Table G.1 shows that **kr-kp** and **car-evaluation** exceed the 100,000-constraint cap of the hybrid solver at every tested depth, while on the remaining full datasets the largest feasible depth D^* rarely exceeds 2 or 3.

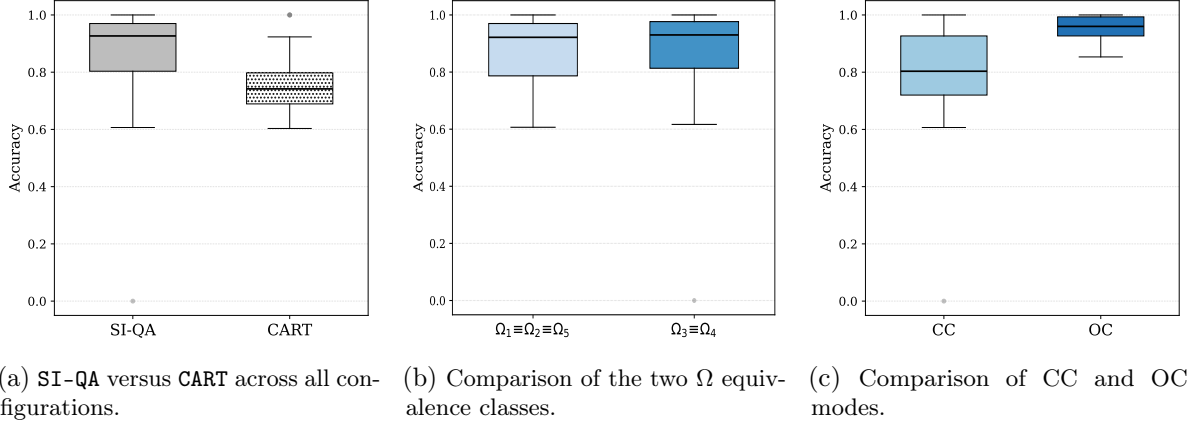


Figure 4: Aggregated accuracy distributions under the subsampled protocol. Accuracy is computed as the number of correctly classified instances divided by the 300-row subsample size. Panel 4a compares **SI-QA** against **CART**, pooling all six random seeds, both classification modes $K \in \{\text{CC}, \text{OC}\}$, both Ω equivalence classes, and all valid depths D . Panel 4b restricts the comparison to **SI-QA** and contrasts the two Ω equivalence classes $\{\Omega_1 \equiv \Omega_2 \equiv \Omega_5\}$ and $\{\Omega_3 \equiv \Omega_4\}$, pooling across modes, seeds, and depths. Panel 4c contrasts the two classification modes for **SI-QA**, pooling across Ω classes, seeds, and depths.

Overall, the QUBO experiments support quantum annealing as an incumbent-generation layer for DNF optimization. In particular, reverse quantum annealing could start from a user-provided **CART** state, even when this solution is not directly admissible as a MILP warm start, and refine it directly in the DNF space, potentially reducing degeneracy while improving solution quality (Deleplanque et al. 2025).

6 Conclusion

This paper develops an optimization framework for Boolean classification that links DT models (Bertsimas and Dunn 2017, Aghaei et al. 2025) and DNF rule-learning models (Dash et al. 2018, Lawless et al. 2023b), to address the rapid growth of the corresponding mathematical programming formulations with respect to the number of features. We address this difficulty by combining equivalence-based reformulations, saturation of the clause space, QUBO encodings, and incremental procedures centered on warm-start reuse.

The theoretical contribution is a unified multi-class model for categorical classification, based on unique class assignment, and ordinal classification, based on first-true logic. Within this model, we compare the alternative treatments of empty and contradictory clauses, prove the corresponding objective-value equivalences, and identify false-satisfiability as a condition under which the contradiction-free and contradiction-allowing formulations coincide. We introduce a saturation principle, in accordance to which every DNF with clauses containing at most D literals admits an equivalent representation whose clauses contain exactly D literals. This reduces the effective clause dictionary from $T_{[\leq D]}$ to

$T_{[=D]}$, provides a common saturated DNF space into which depth- D decision trees can be embedded, and removes unnecessary variables and penalties from the QUBO formulation.

These reductions motivate the single- and double-incremental learning procedures. Rather than solving one large model from scratch, the algorithms solve a sequence of increasingly expressive problems, lifting incumbents obtained from previous depths, **CART**, or quantum annealing. The computational results on seven benchmark datasets show that this strategy is effective for both categorical and ordinal classification. The best-DNF envelope improves on non-incremental **FlowOCT** by 18.8% in normalized objective while being 35.1% faster, and improves on warm-started **FlowOCT** by 3.6% in objective while being 46.7% faster. Quantum annealing also acts as a useful incumbent generator, improving over the **CART** warm start by 18.8 correctly classified instances on average in categorical classification and by 61.4 in ordinal classification.

Future work should strengthen the weak continuous relaxation identified in Proposition 1, develop decomposition methods for larger instances, including the integration of the proposed incremental learning procedure within the column-generation scheme of Dash et al. (2018), Lawless et al. (2023a), and further exploit saturation-based reductions as quantum hardware improves. The same framework also points toward fair, robust and prescriptive extensions of Boolean classification. Overall, the paper shows that exact Boolean classifiers can remain interpretable and computationally competitive when structural reductions and warm-start mechanisms are fully exploited and integrated into the optimization pipeline.

References

- Aghaei, S., A. Gómez, and P. Vayanos. 2025. Strong optimal classification trees. *Operations Research* 73 (4): 2223–2241.
- Albash, T., and D. A. Lidar. 2018. Adiabatic quantum computation. *Reviews of Modern Physics* 90 (1): 015002.
- Bernal, D. E., K. E. Booth, R. Dridi, H. Alghassi, S. Tayur, and D. Venturelli. 2020. Integer programming techniques for minor-embedding in quantum annealers. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings 17*, 112–129. Springer.
- Bertsimas, D., and J. Dunn. 2017. Optimal classification trees. *Machine Learning* 106 (7): 1039–1082.
- Bertsimas, D., and R. Shioda. 2007. Classification and Regression via Integer Optimization. *Operations Research* 55 (2): 252–271.
- Billionnet, A., and S. Elloumi. 2007. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical programming* 109 (1): 55–68.

- Blanquero, R., E. Carrizosa, C. Molero-Río, and D. R. Morales. 2021. Optimal randomized classification trees. *Computers & Operations Research* 132:105281.
- Blanquero, R., E. Carrizosa, C. Molero-Río, and D. Romero Morales. 2020. Sparsity in optimal randomized classification trees. *European Journal of Operational Research* 284 (1): 255–272.
- Boros, E., Y. Crama, P. L. Hammer, T. Ibaraki, A. Kogan, and K. Makino. 2011. Logical analysis of data: classification with justification. *Annals of Operations Research* 188 (1): 33–61.
- Bouhmala, N. 2019. Combining simulated annealing with local search heuristic for MAX-SAT. *Journal of Heuristics* 25 (1): 47–69.
- Chen, L.-Y., and S. Lee. 2018. Best subset binary prediction. *Journal of Econometrics* 206 (1): 39–56.
- Ciocan, D. F., and V. V. Mišić. 2022. Interpretable Optimal Stopping. *Management Science* 68 (3): 1616–1638.
- Clark, P., and T. Niblett. 1989. The CN2 induction algorithm. *Machine learning* 3 (4): 261–283.
- Dash, S., O. Gunluk, and D. Wei. 2018. Boolean decision rules via column generation. *Advances in neural information processing systems* 31.
- Deleplanque, S., L. F. Pérez Armas, and S. Creemers. 2025. solQHealer: Quantum Procedures for Rendering Infeasible Solutions Feasible: A Proof of Concept with the Maximum Independent Set Problem and 3-SAT: S. Deleplanque et al. *Journal of Heuristics* 31 (3): 30.
- Hammer, P. L., and T. O. Bonates. 2006. Logical analysis of data—An overview: From combinatorial optimization to medical applications. *Annals of Operations Research* 148 (1): 203–225.
- Heath, D., S. Kasif, and S. Salzberg. 1993. Induction of oblique decision trees. In *IJCAI*, Volume 1993, 1002–1007. Citeseer.
- Hu, X., C. Rudin, and M. Seltzer. 2019. Optimal Sparse Decision Trees. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 7267–7275.
- Janota, M., and A. Morgado. 2020. SAT-Based Encodings for Optimal Decision Trees with Explicit Paths. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 501–518.
- Kadowaki, T., and H. Nishimori. 1998. Quantum annealing in the transverse Ising model. *Physical Review E* 58 (5): 5355.
- Karp, R. M. 2009. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, 219–241. Springer.
- Kelly, M., R. Longjohn, and K. Nottingham. 2023. The UCI Machine Learning Repository. <https://archive.ics.uci.edu>.
- Kirkpatrick, S., C. D. Gelatt Jr, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220 (4598): 671–680.
- Lawless, C., S. Dash, O. Gunluk, and D. Wei. 2023a. Interpretable and fair boolean rule sets via column generation. *Journal of Machine Learning Research* 24 (229): 1–50.

- Lawless, C., S. Dash, O. Gunluk, and D. Wei. 2023b. Interpretable and fair boolean rule sets via column generation. *Journal of Machine Learning Research* 24 (229): 1–50.
- Liu, Y. 2022. bsnsing: A decision tree induction method based on recursive optimal boolean rule composition. *INFORMS Journal on Computing* 34 (6): 2908–2929.
- Loh, W.-Y. 2014. Fifty years of classification and regression trees. *International Statistical Review* 82 (3): 329–348.
- Lucas, A. 2014. Ising formulations of many NP problems. *Frontiers in physics* 2:5.
- Marchand, M., and J. Shawe-Taylor. 2002. The set covering machine. *Journal of Machine Learning Research* 3 (Dec): 723–746.
- Mišić, V. V. 2020. Optimization of Tree Ensembles. *Operations Research* 68 (5): 1605–1624.
- Narodytska, N., A. Ignatiev, F. Pereira, and J. Marques-Silva. 2018. Learning Optimal Decision Trees with SAT. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 1362–1368.
- Nijssen, S., and E. Fromont. 2010. Optimal Constraint-Based Decision Tree Induction from Itemset Lattices. *Data Mining and Knowledge Discovery* 21 (1): 9–51.
- Pagallo, G. 1989. Learning DNF by Decision Trees. In *IJCAI*, Volume 89, 639–644.
- Schneeweiss, W. G. 2012. *Boolean functions: with engineering applications and computer programs*. Springer Science & Business Media.
- Serra, T., T. Huang, A. U. Raghunathan, and D. Bergman. 2022. Template-based minor embedding for adiabatic quantum optimization. *INFORMS Journal on Computing* 34 (1): 427–439.
- Üney, F., and M. Türkay. 2006. A mixed-integer programming approach to multi-class data classification problem. *European journal of operational research* 173 (3): 910–920.
- Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 27 (11): 1134–1142.
- Verhaeghe, H., S. Nijssen, G. Pesant, C.-G. Quimper, and P. Schaus. 2020. Learning Optimal Decision Trees Using Constraint Programming. *Constraints* 25 (3-4): 226–250.
- Verwer, S., and Y. Zhang. 2017. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 94–103. Springer.
- Verwer, S., and Y. Zhang. 2019. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI conference on artificial intelligence*, Volume 33, 1625–1632.
- Zhu, H., P. Murali, D. Phan, L. Nguyen, and J. Kalagnanam. 2020. A scalable mip-based method for learning optimal multivariate decision trees. *Advances in neural information processing systems* 33:1771–1781.

Competing interests

The authors declare no competing interests.

A Empty and contradictory clauses reformulations

In formulation (4), the sets $\Omega_1, \dots, \Omega_5$ encode alternative algebraic treatments of empty and contradictory clauses. Recall that each clause is a conjunction of selected literals. Therefore: (i) an empty clause (no selected literals) evaluates to the Boolean constant **true**, i.e., it is satisfied by every instance; and (ii) a contradictory clause (some feature appears both affirmed and negated) evaluates to the Boolean constant **false**, i.e., it is satisfied by no instance. The two cases are controlled by constraining the literal-selection variables $\bar{\phi}_{f,c}^t$ and $\underline{\phi}_{f,c}^t$.

A.1 Four basic specifications: Ω_1 – Ω_4

The first four sets are expressed as:

$$\Omega_1 = \left\{ \varepsilon_{t,c} = 0, \quad t \in \mathcal{T}, c \in \mathcal{C} \right. \quad (\text{A.1a})$$

$$\Omega_2 = \begin{cases} \varepsilon_{t,c} = 0, & t \in \mathcal{T}, c \in \mathcal{C}, \\ \sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right) \geq 1, & t \in \mathcal{T}, c \in \mathcal{C} \end{cases} \quad (\text{A.1b})$$

$$\Omega_3 = \begin{cases} \varepsilon_{t,c} = 0, & t \in \mathcal{T}, c \in \mathcal{C}, \\ \bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \leq 1, & t \in \mathcal{T}, c \in \mathcal{C}, f \in \mathcal{F} \end{cases} \quad (\text{A.1c})$$

$$\Omega_4 = \Omega_2 \cap \Omega_3. \quad (\text{A.1d})$$

Their logical meaning is immediate: Ω_1 imposes no restriction on emptiness or contradiction (it only fixes $\varepsilon_{t,c}$ to zero, hence it does not neutralize emptiness); Ω_2 forbids empty clauses by enforcing at least one selected literal in each (t, c) ; Ω_3 forbids contradictions by preventing the simultaneous selection of both polarities of the same feature, but it still allows empty clauses; Ω_4 forbids both empty and contradictory clauses.

A.2 Why empty clauses dominate under Ω_1 and Ω_3

Assume that, for some pair (t, c) , clause t is empty, i.e., $\bar{\phi}_{f,c}^t = \underline{\phi}_{f,c}^t = 0$ for all $f \in \mathcal{F}$. Then, by the clause-definition constraints (4b)–(4c), all feature-dependent terms vanish and the clause is satisfied by every instance:

$$\gamma_{i,t,c} = 1 \quad \forall i \in \mathcal{I}. \quad (\text{A.2})$$

Consequently, the DNF aggregation constraints (4e)–(4f) force

$$\tilde{\gamma}_{i,c} = 1 \quad \forall i \in \mathcal{I}, \quad (\text{A.3})$$

regardless of the values of $\gamma_{i,t',c}$ for all other $t' \neq t$. In other words, a single empty clause makes the whole class- c DNF identically true, thereby overwriting the contribution of all non-empty clauses associated with the same class.

This explains the behavior of Ω_1 and Ω_3 : they allow empty clauses, and any empty clause immediately dominates the truth value of $\tilde{\gamma}_{i,c}$ through (A.3). From a modeling perspective, this may look undesirable; however, under the objective (4a), an empty clause typically yields many false positives for class c (since (A.3) holds for all i), and therefore it is generally suboptimal unless the data are degenerate (e.g., all instances belong to class c or the classification rule is otherwise trivial). By contrast, Ω_2 and Ω_4 preclude this phenomenon by construction.

A.3 Neutralizing empty clauses: Ω_5

We now consider a formulation in which empty clauses are allowed but are treated as inactive, i.e., they do not contribute to the Boolean value of the DNF. To do so we introduce binary indicators $\varepsilon_{t,c}$ that detect emptiness and then remove empty clauses from the OR aggregation. We define:

$$\Omega_5 = \begin{cases} \varepsilon_{t,c} \leq \frac{1}{F} \sum_{f \in \mathcal{F}} (1 - \bar{\phi}_{f,c}^t - \underline{\phi}_{f,c}^t) & t \in \mathcal{T}, c \in \mathcal{C} & (\text{A.4a}) \\ \varepsilon_{t,c} \geq \gamma_{i,t,c} - \sum_{f \in \mathcal{F}} (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) & i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C} & (\text{A.4b}) \\ \sum_{t \in \mathcal{T}} \varepsilon_{t,c} \leq T - 1 & c \in \mathcal{C} & (\text{A.4c}) \\ \varepsilon_{t,c} \in \{0, 1\} & t \in \mathcal{T}, c \in \mathcal{C}. & (\text{A.4d}) \end{cases}$$

Constraints (A.4a)–(A.4b) enforce

$$\varepsilon_{t,c} = 1 \iff \sum_{f \in \mathcal{F}} (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) = 0,$$

i.e., $\varepsilon_{t,c}$ acts as an emptiness indicator. Indeed, if clause (t, c) is empty then the RHS of (A.4a) equals 1, and by (A.2) we have $\gamma_{i,t,c} = 1$ for all i , so (A.4b) yields $\varepsilon_{t,c} \geq 1$, hence $\varepsilon_{t,c} = 1$. Conversely, if at least one literal is selected then $\sum_f (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) \geq 1$, implying that the RHS of (A.4a) is strictly smaller than 1, and with $\varepsilon_{t,c} \in \{0, 1\}$ this forces $\varepsilon_{t,c} = 0$. Constraint (A.4c) prevents the degenerate

case where all clauses for a given class are empty (so that at least one non-empty clause remains available to define the class rule). Under Ω_5 , the class-level OR is evaluated over effective clause contributions, so that an empty clause does not activate $\tilde{\gamma}_{i,c}$. Equivalently, introduce auxiliary binary variables $\hat{\gamma}_{i,t,c}$ to represent the effective contribution of clause (t, c) :

$$\hat{\gamma}_{i,t,c} \leq \gamma_{i,t,c}, \quad i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}, \quad (\text{A.5a})$$

$$\hat{\gamma}_{i,t,c} \leq 1 - \varepsilon_{t,c}, \quad i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}, \quad (\text{A.5b})$$

$$\hat{\gamma}_{i,t,c} \geq \gamma_{i,t,c} - \varepsilon_{t,c}, \quad i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}, \quad (\text{A.5c})$$

$$\hat{\gamma}_{i,t,c} \in \{0, 1\}, \quad i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}. \quad (\text{A.5d})$$

These constraints linearize $\hat{\gamma}_{i,t,c} = \gamma_{i,t,c}(1 - \varepsilon_{t,c})$. Then, replace (4e)–(4f) with:

$$\tilde{\gamma}_{i,c} \leq \sum_{t \in \mathcal{T}} \hat{\gamma}_{i,t,c}, \quad i \in \mathcal{I}, c \in \mathcal{C}, \quad (\text{A.6a})$$

$$\tilde{\gamma}_{i,c} \geq \hat{\gamma}_{i,t,c}, \quad i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}. \quad (\text{A.6b})$$

Therefore, if $\gamma_{i,t,c} = 1$ *only because clause (t, c) is empty*, then $\varepsilon_{t,c} = 1$ and (A.5b) forces $\hat{\gamma}_{i,t,c} = 0$, so $\tilde{\gamma}_{i,c}$ is no longer forced to 1. In short: Ω_5 permits empty clauses while ensuring they do not dominate the DNF truth value.

A.4 Empty and contradictory clauses reformulations in the QUBO

For the formulation-specific component of Q_M^K , we consider the three specifications corresponding to (7), (8), and both combined. The corresponding penalty component $\tilde{Q}_M$ is given by

$$\tilde{Q}_2 = \lambda^{(2)} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \left(\sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right) - 1 - s_{t,c}^{(2,2)} \right)^2,$$

$$\tilde{Q}_3 = \lambda^{(3)} \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} \left(\bar{\phi}_{f,c}^t \underline{\phi}_{f,c}^t \right),$$

and

$$\tilde{Q}_4 = \lambda^{(4,1)} \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \left(\sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right) - 1 - s_{t,c}^{(4,2)} \right)^2 + \lambda^{(4,2)} \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \sum_{c \in \mathcal{C}} \left(\bar{\phi}_{f,c}^t \underline{\phi}_{f,c}^t \right).$$

Here, the penalties $\tilde{Q}_2$, $\tilde{Q}_3$, and $\tilde{Q}_4$ respectively enforce nonempty clauses, non-contradictory clauses, and both restrictions. The slack variables $s_{t,c}^{(2,2)}$ and $s_{t,c}^{(4,2)}$ are nonnegative integer variables encoded

in binary. For Ω_1 , one simply takes $\tilde{Q}_1 = 0$, apart from the fixing of $\varepsilon_{t,c} = 0$ when ε variables are retained. The construction can also be extended to Ω_5 by adding penalties for the defining constraints (A.4a)–(A.4c), although in our implementation the QUBO warm-start generation focuses on the nonempty and/or non-contradictory specifications.

B : Mathematical proofs

B.1 Auxiliary lemmas

Lemma 2. Let ϕ denote the binary array of the form $\{(\bar{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t)\}_{f \in \mathcal{F}, c \in \mathcal{C}, t \in \mathcal{T}}$, and define $e(\phi)$ and $g(\phi)$ as follows:

$$g_{i,t,c}(\phi) = \prod_{f: \bar{\phi}_{f,c}^t = 1} v_{i,f} \cdot \prod_{f: \underline{\phi}_{f,c}^t = 1} (1 - v_{i,f}) \quad \text{and} \quad e_{t,c}(\phi) = \mathbf{1} \left\{ \sum_{f \in \mathcal{F}} (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) = 0 \right\}.$$

If ϕ and γ satisfy constraints (4b) and (4c), then $\gamma_{i,t,c} = g_{i,t,c}(\phi)$. If ϕ , γ and ε satisfy constraints (4b)–(4c) and (9a)–(9b), then $\varepsilon_{t,c} = e_{t,c}(\phi)$. Hence, γ and ε are functions of ϕ .

Proof. To prove $\gamma_{i,t,c} = g_{i,t,c}(\phi)$, for a given (i, t, c) , let's discuss three cases. To prove $\gamma_{i,t,c} = g_{i,t,c}(\phi)$, for a given (i, t, c) , let's discuss three cases.

- Case 1. There exists a $f \in \mathcal{F}$ such that $v_{i,f} = \underline{\phi}_{f,c}^t = 1$. By constraint (4b), $\gamma_{i,t,c} = 0$. Meanwhile, $g_{i,t,c}(\phi) = 0$ because it is a product containing the term $(1 - v_{i,f}) = 0$.
- Case 2. There exists a $f \in \mathcal{F}$ such that $1 - v_{i,f} = \bar{\phi}_{f,c}^t = 1$. By constraint (4b), $\gamma_{i,t,c} = 0$. Meanwhile, $g_{i,t,c}(\phi) = 0$ because it is a product containing the term $v_{i,f} = 0$.
- Case 3. For all $f \in \mathcal{F}$ whenever $v_{i,f} = 1$, $\underline{\phi}_{f,c}^t = 0$ and whenever $v_{i,f} = 0$, $\bar{\phi}_{f,c}^t = 1$. In this case, the right hand side of constraints (4b) and (4c) both reduce to 1, forcing $\gamma_{i,t,c} = 1$. Meanwhile, the product in $g_{i,t,c}(\phi)$ consists of only 1's.

Therefore, under all cases, $\gamma_{i,t,c} = g_{i,t,c}(\phi)$.

To prove $\varepsilon_{t,c} = e_{t,c}(\phi)$, we demonstrate that $\varepsilon_{t,c} = 1$ iff $\sum_{f \in \mathcal{F}} (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) = 0$. We discuss two cases :

- Case 1. If for each $f \in \mathcal{F}$; $\bar{\phi}_{f,c}^t = \underline{\phi}_{f,c}^t = 0$, then by constraints (4b) and (4c), $\gamma_{i,t,c} = 1$. It follows that constraints on ε in (9a) and (9b) become

$$\varepsilon_{t,c} \leq \frac{1}{F} \sum_{f \in \mathcal{F}} 1 = 1 \quad \text{and} \quad \varepsilon_{t,c} \geq 1 - \sum_{f \in \mathcal{F}} 0 = 1.$$

Therefore $\varepsilon_{t,c} = 1$.

- Case 2. Otherwise, there exists $f \in \mathcal{F}$ for which $\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \geq 1$, so (9a) becomes

$$\varepsilon_{t,c} \leq \frac{1}{F} \sum_{f \in \mathcal{F}} (1 - \bar{\phi}_{f,c}^t - \underline{\phi}_{f,c}^t) \leq \frac{F-1}{F} < 1.$$

Therefore $\varepsilon_{t,c} = 0$.

□

B.2 Main theorems and propositions

Theorem 1

Proof. Fix $K \in \{\text{CC}, \text{OC}\}$. Since $\Omega_2 \subseteq \Omega_1$, every feasible solution of problem (3) $B_{M=2}^K(D)$ is feasible for problem (3) $B_{M=1}^K(D)$. Hence, $B_1^K(D) \geq B_2^K(D)$. We prove the reverse inequality by showing that every optimal solution of $B_1^K(D)$ can be transformed into a feasible solution of $B_2^K(D)$ with the same objective value.

Let $(\zeta, \gamma, \tilde{\gamma}, \bar{\phi}, \underline{\phi}, \varepsilon)$ be an optimal solution of problem (3) $B_{M=1}^K(D)$. Since this solution satisfies Ω_1 , we have $\varepsilon_{t,c} = 0$, for all $t \in \mathcal{T}$, $c \in \mathcal{C}$. Define the set of classes containing at least one empty clause as

$$\mathcal{C}_\emptyset := \left\{ c \in \mathcal{C} : \exists t \in \mathcal{T} \text{ such that } \bar{\phi}_{f,c}^t = \underline{\phi}_{f,c}^t = 0, \forall f \in \mathcal{F} \right\}.$$

If $\mathcal{C}_\emptyset = \emptyset$, then every clause of the optimal solution is nonempty. Since the solution already satisfies Ω_1 , and since the only additional requirement in Ω_2 is the nonemptiness of every clause, the same solution is feasible for $B_{M=2}^K(D)$. Therefore, in this case, $B_1^K(D) \leq B_2^K(D)$.

It remains to consider the case $\mathcal{C}_\emptyset \neq \emptyset$. By the standing assumptions, choose one feature $f^0 \in \mathcal{F}$ and one clause index $t^+ \in \mathcal{T}$ such that $\mathcal{T} \setminus \{t^+\} \neq \emptyset$. We construct new literal-selection variables $(\bar{\phi}', \underline{\phi}')$ as follows. For every $c \notin \mathcal{C}_\emptyset$, keep the original clauses:

$$\bar{\phi}_{f,c}'^t = \bar{\phi}_{f,c}^t, \quad \underline{\phi}_{f,c}'^t = \underline{\phi}_{f,c}^t, \quad \forall f \in \mathcal{F}, t \in \mathcal{T}.$$

For every $c \in \mathcal{C}_\emptyset$, replace the whole class- c DNF by the tautological disjunction of the two polarities of feature f^0 :

$$\bar{\phi}_{f,c}'^t = \begin{cases} 1, & \text{if } f = f^0 \text{ and } t = t^+, \\ 0, & \text{otherwise,} \end{cases} \quad \underline{\phi}_{f,c}'^t = \begin{cases} 1, & \text{if } f = f^0 \text{ and } t \neq t^+, \\ 0, & \text{otherwise.} \end{cases}$$

Define $\varepsilon'_{t,c} := 0$, for all $t \in \mathcal{T}$, $c \in \mathcal{C}$. For every $i \in \mathcal{I}$, $t \in \mathcal{T}$, and $c \in \mathcal{C}$, define

$$\gamma'_{i,t,c} := g_{i,t,c}(\bar{\phi}', \underline{\phi}') = f_{v_i}(\bar{\phi}_{\cdot,c}'^t, \underline{\phi}_{\cdot,c}'^t),$$

where $g_{i,t,c}$ denotes the Boolean value of clause (t, c) on observation i . Finally, define

$$\tilde{\gamma}'_{i,c} := \max_{t \in \mathcal{T}} \gamma'_{i,t,c}, \quad \forall i \in \mathcal{I}, c \in \mathcal{C},$$

and set $\zeta'_i := \zeta_i$, for all $i \in \mathcal{I}$. We now verify that $(\zeta', \gamma', \tilde{\gamma}', \overline{\phi}', \underline{\phi}', \epsilon')$ is feasible for problem (3) $B_{M=2}^K(D)$. First, by construction of γ' , constraints (4b)–(4c) are satisfied. Since $\epsilon'_{t,c} = 0$ for all (t, c) and

$$\tilde{\gamma}'_{i,c} = \max_{t \in \mathcal{T}} \gamma'_{i,t,c},$$

the aggregation constraints defining $\tilde{\gamma}'$ are also satisfied. Second, the cardinality constraint is satisfied. If $c \notin \mathcal{C}_\emptyset$, then the clauses are unchanged from the original feasible solution. If $c \in \mathcal{C}_\emptyset$, then every clause contains exactly one literal, namely either the f^0 -literal with the overline polarity or the f^0 -literal with the underline polarity. Hence, for every $t \in \mathcal{T}$ and $c \in \mathcal{C}_\emptyset$,

$$\sum_{f \in \mathcal{F}} \left(\overline{\phi}'_{f,c}{}^t + \underline{\phi}'_{f,c}{}^t \right) = 1 \leq D.$$

Third, the solution satisfies Ω_2 . Indeed, $\epsilon'_{t,c} = 0$ for all (t, c) . Moreover, if $c \notin \mathcal{C}_\emptyset$, then no clause of class c was empty in the original solution, so

$$\sum_{f \in \mathcal{F}} \left(\overline{\phi}'_{f,c}{}^t + \underline{\phi}'_{f,c}{}^t \right) \geq 1, \quad \forall t \in \mathcal{T}.$$

If $c \in \mathcal{C}_\emptyset$, then the preceding construction gives

$$\sum_{f \in \mathcal{F}} \left(\overline{\phi}'_{f,c}{}^t + \underline{\phi}'_{f,c}{}^t \right) = 1, \quad \forall t \in \mathcal{T}.$$

Thus every clause is nonempty.

It remains to show that the class-level truth values are preserved on the training sample. If $c \notin \mathcal{C}_\emptyset$, then the clauses of class c are unchanged, so $\gamma'_{i,t,c} = \gamma_{i,t,c}$, for all $i \in \mathcal{I}$, $t \in \mathcal{T}$, and therefore $\tilde{\gamma}'_{i,c} = \tilde{\gamma}_{i,c}$. If $c \in \mathcal{C}_\emptyset$, then by definition there exists $t_c \in \mathcal{T}$ such that

$$\overline{\phi}_{f,c}{}^{t_c} = \underline{\phi}_{f,c}{}^{t_c} = 0, \quad \forall f \in \mathcal{F}.$$

Since $\epsilon_{t,c} = 0$ under Ω_1 , constraints (4b)–(4c) imply

$$\gamma_{i,t_c,c} = 1, \quad \forall i \in \mathcal{I}.$$

Hence the aggregation constraints imply

$$\tilde{\gamma}_{i,c} = 1, \quad \forall i \in \mathcal{I}.$$

On the other hand, in the constructed solution, class c contains both polarities of feature f^0 across its clauses. Therefore, for every $i \in \mathcal{I}$, either the clause indexed by t^+ is true or at least one clause indexed by $t \neq t^+$ is true. Consequently, $\tilde{\gamma}'_{i,c} = 1 = \tilde{\gamma}_{i,c}$, so that $\tilde{\gamma}'_{i,c} = \tilde{\gamma}_{i,c}$, for every $i \in \mathcal{I}$ and $c \in \mathcal{C}$. Since $\zeta' = \zeta$ and $\tilde{\gamma}' = \tilde{\gamma}$ on the training sample, all constraints involving correctness and class logic remain satisfied. In particular, the constraints

$$\zeta_i \leq \tilde{\gamma}_{i,u_i}, \quad \forall i \in \mathcal{I},$$

remain valid after replacing $(\zeta, \tilde{\gamma})$ by $(\zeta', \tilde{\gamma}')$. Moreover, the constraints in Γ^{CC} and Γ^{OC} depend only on ζ and $\tilde{\gamma}$, and are therefore preserved for both $K = CC$ and $K = OC$.

We have constructed a feasible solution of problem (3) $B_{M=2}^K(D)$ with objective value

$$\sum_{i \in \mathcal{I}} \zeta'_i = \sum_{i \in \mathcal{I}} \zeta_i.$$

Since the original solution was optimal for $B_{M=1}^K(D)$, this proves

$$B_1^K(D) \leq B_2^K(D).$$

Combining this inequality with $B_1^K(D) \geq B_2^K(D)$ gives

$$B_1^K(D) = B_2^K(D).$$

□

Theorem 2

Proof. Fix $K \in \{CC, OC\}$. Throughout the proof, we use the standing assumptions $|\mathcal{F}| \geq 1$, $|\mathcal{T}| \geq 2$, and $1 \leq D \leq F$, where $F = |\mathcal{F}|$ and $T = |\mathcal{T}|$. We first prove that $B_5^K(D) \leq B_2^K(D)$.

Let $(\zeta, \gamma, \tilde{\gamma}, \bar{\phi}, \underline{\phi}, \varepsilon)$ be an arbitrary feasible solution of problem (3) $B_{M=5}^K(D)$. Define $\phi := (\bar{\phi}, \underline{\phi})$ and, for every $(t, c) \in \mathcal{T} \times \mathcal{C}$, define the number of selected literals in clause (t, c) by

$$S_{t,c} := \sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right).$$

By Lemma 2, applied to the feasible solution of $B_{M=5}^K(D)$,

$$\varepsilon_{t,c} = 1 \iff S_{t,c} = 0, \quad \forall t \in \mathcal{T}, c \in \mathcal{C}.$$

Moreover, constraint (9c) implies

$$\sum_{t \in \mathcal{T}} \varepsilon_{t,c} \leq T - 1, \quad \forall c \in \mathcal{C}.$$

Hence, for every class $c \in \mathcal{C}$, there exists at least one clause index $t_c^* \in \mathcal{T}$ such that $\varepsilon_{t_c^*,c} = 0$. By the preceding equivalence, this clause is nonempty, that is, $S_{t_c^*,c} \geq 1$.

We now construct a new solution by replacing each empty clause by a copy of the nonempty clause t_c^* of the same class. For every $f \in \mathcal{F}$, $t \in \mathcal{T}$, and $c \in \mathcal{C}$, define

$$\bar{\phi}_{f,c}'^t = \begin{cases} \bar{\phi}_{f,c}^t, & \text{if } \varepsilon_{t,c} = 0, \\ \bar{\phi}_{f,c}^{t_c^*}, & \text{if } \varepsilon_{t,c} = 1, \end{cases} \quad \underline{\phi}_{f,c}'^t = \begin{cases} \underline{\phi}_{f,c}^t, & \text{if } \varepsilon_{t,c} = 0, \\ \underline{\phi}_{f,c}^{t_c^*}, & \text{if } \varepsilon_{t,c} = 1. \end{cases}$$

Let $\phi' := (\bar{\phi}', \underline{\phi}')$. For every $i \in \mathcal{I}$, $t \in \mathcal{T}$, and $c \in \mathcal{C}$, define $\gamma'_{i,t,c} := g_{i,t,c}(\phi')$. For every $i \in \mathcal{I}$ and $c \in \mathcal{C}$, define $\tilde{\gamma}'_{i,c} := \max\{\gamma'_{i,t,c} : t \in \mathcal{T}\}$. Finally, set $\zeta'_i := \zeta_i$, for every $i \in \mathcal{I}$, and $\varepsilon'_{t,c} := 0$, for all $t \in \mathcal{T}$, $c \in \mathcal{C}$.

We verify that $(\zeta', \gamma', \tilde{\gamma}', \bar{\phi}', \underline{\phi}', \varepsilon')$ is feasible for problem (3) $B_{M=2}^K(D)$.

First, the clause-evaluation constraints are satisfied. Indeed, for every $i \in \mathcal{I}$, $t \in \mathcal{T}$, $f \in \mathcal{F}$, and $c \in \mathcal{C}$, define

$$m'_{i,t,f,c} := (1 - v_{i,f})\bar{\phi}_{f,c}'^t + v_{i,f}\underline{\phi}_{f,c}'^t.$$

By definition of f_{v_i} , we have $\gamma'_{i,t,c} = 1$ if and only if $m'_{i,t,f,c} = 0$ for all $f \in \mathcal{F}$. Hence,

$$\gamma'_{i,t,c} \leq 1 - m'_{i,t,f,c}, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, f \in \mathcal{F}, c \in \mathcal{C},$$

and

$$\gamma'_{i,t,c} \geq 1 - \sum_{f \in \mathcal{F}} m'_{i,t,f,c}, \quad \forall i \in \mathcal{I}, t \in \mathcal{T}, c \in \mathcal{C}.$$

Thus constraints (4b)–(4c) are satisfied.

Second, the cardinality constraint is satisfied. Define

$$S'_{t,c} := \sum_{f \in \mathcal{F}} (\bar{\phi}_{f,c}'^t + \underline{\phi}_{f,c}'^t).$$

If $\varepsilon_{t,c} = 0$, then clause (t, c) is unchanged, so $S'_{t,c} = S_{t,c} \leq D$, because the original solution is feasible for $B_{M=5}^K(D)$. If $\varepsilon_{t,c} = 1$, then clause (t, c) is replaced by clause (t_c^*, c) , and therefore $S'_{t,c} = S_{t_c^*,c} \leq D$, again because the original solution satisfies the cardinality constraint. Hence

$$S'_{t,c} \leq D, \quad \forall t \in \mathcal{T}, c \in \mathcal{C}.$$

Third, every clause is nonempty. If $\varepsilon_{t,c} = 0$, then Lemma 2 gives $S_{t,c} \geq 1$, and thus $S'_{t,c} = S_{t,c} \geq 1$. If $\varepsilon_{t,c} = 1$, then $S'_{t,c} = S_{t_c^*,c} \geq 1$. Therefore,

$$\sum_{f \in \mathcal{F}} (\bar{\phi}'_{f,c} + \underline{\phi}'_{f,c}) \geq 1, \quad \forall t \in \mathcal{T}, c \in \mathcal{C}.$$

Together with $\varepsilon'_{t,c} = 0$ for all (t, c) , this proves that the constructed solution satisfies Ω_2 .

Fourth, the aggregation constraints are satisfied. Since $\varepsilon'_{t,c} = 0$ for all (t, c) and

$$\tilde{\gamma}'_{i,c} = \max_{t \in \mathcal{T}} \gamma'_{i,t,c},$$

constraints (4f)–(4e) hold for $(\gamma', \tilde{\gamma}', \varepsilon')$.

It remains to show that the class-level truth values are preserved on the training sample. Fix $i \in \mathcal{I}$ and $c \in \mathcal{C}$. In the original feasible solution of $B_{M=5}^K(D)$, an empty clause satisfies $\varepsilon_{t,c} = 1$ and $\gamma_{i,t,c} = 1$, hence $\gamma_{i,t,c} - \varepsilon_{t,c} = 0$ for every empty clause. For a nonempty clause, $\varepsilon_{t,c} = 0$, so $\gamma_{i,t,c} - \varepsilon_{t,c} = \gamma_{i,t,c}$. Thus, the aggregation constraints imply

$$\tilde{\gamma}_{i,c} = 1 \iff \exists t \in \mathcal{T} \text{ such that } \gamma_{i,t,c} = 1 \text{ and } \varepsilon_{t,c} = 0.$$

On the other hand, by construction,

$$\begin{cases} \gamma'_{i,t,c} = \gamma_{i,t,c} & \text{whenever } \varepsilon_{t,c} = 0, \\ \gamma'_{i,t,c} = \gamma_{i,t_c^*,c} & \text{whenever } \varepsilon_{t,c} = 1. \end{cases}$$

Since $\varepsilon_{t_c^*,c} = 0$, it follows that

$$\exists t \in \mathcal{T} \text{ such that } \gamma'_{i,t,c} = 1 \iff \exists t \in \mathcal{T} \text{ such that } \gamma_{i,t,c} = 1 \text{ and } \varepsilon_{t,c} = 0.$$

Therefore, $\tilde{\gamma}'_{i,c} = \tilde{\gamma}_{i,c}$, for all, $i \in \mathcal{I}$, $c \in \mathcal{C}$. Since

$$\zeta' = \zeta \quad \text{and} \quad \tilde{\gamma}' = \tilde{\gamma},$$

all correctness constraints and all constraints in Γ^K remain satisfied. Hence the constructed solution is feasible for problem (3) $B_{M=2}^K(D)$ and has objective value

$$\sum_{i \in \mathcal{I}} \zeta'_i = \sum_{i \in \mathcal{I}} \zeta_i.$$

Since the original feasible solution of $B_{M=5}^K(D)$ was arbitrary, we obtain $B_5^K(D) \leq B_2^K(D)$.

We now prove the reverse inequality $B_2^K(D) \leq B_5^K(D)$. Let $(\zeta, \gamma, \tilde{\gamma}, \bar{\phi}, \underline{\phi}, \varepsilon)$ be an arbitrary feasible solution of problem (3) $B_{M=2}^K(D)$. Since this solution satisfies Ω_2 , we have $\varepsilon_{t,c} = 0$, for all $t \in \mathcal{T}$, $c \in \mathcal{C}$, and

$$S_{t,c} := \sum_{f \in \mathcal{F}} \left(\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t \right) \geq 1, \quad \forall t \in \mathcal{T}, c \in \mathcal{C}.$$

Moreover, the cardinality constraint gives $S_{t,c} \leq D$, for all $t \in \mathcal{T}$, $c \in \mathcal{C}$. Since $D \leq F$, we have

$$1 \leq S_{t,c} \leq F, \quad \forall t \in \mathcal{T}, c \in \mathcal{C}.$$

We show that the same variables are feasible for $B_{M=5}^K(D)$. The constraints (4b)–(4c), (4d), (4f)–(4e), the correctness constraints, and the constraints in Γ^K are unchanged. It remains only to verify the defining constraints of Ω_5 . For every $(t, c) \in \mathcal{T} \times \mathcal{C}$,

$$\frac{1}{F} \sum_{f \in \mathcal{F}} \left(1 - \bar{\phi}_{f,c}^t - \underline{\phi}_{f,c}^t \right) = 1 - \frac{S_{t,c}}{F} \geq 0,$$

because $S_{t,c} \leq F$. Since $\varepsilon_{t,c} = 0$, the upper-bound constraint in Ω_5 is satisfied. Furthermore, for every $i \in \mathcal{I}$,

$$\gamma_{i,t,c} - S_{t,c} \leq 1 - S_{t,c} \leq 0 = \varepsilon_{t,c},$$

because $\gamma_{i,t,c} \leq 1$ and $S_{t,c} \geq 1$. Hence the lower-bound constraint in Ω_5 is satisfied. Finally,

$$\sum_{t \in \mathcal{T}} \varepsilon_{t,c} = 0 \leq T - 1, \quad \forall c \in \mathcal{C},$$

and the binary restriction on ε is also satisfied. Therefore the same solution is feasible for problem (3) $B_{M=5}^K(D)$ with the same objective value. Since the feasible solution of $B_{M=2}^K(D)$ was arbitrary, $B_2^K(D) \leq B_5^K(D)$. Combining the two inequalities gives $B_5^K(D) = B_2^K(D)$. By Theorem 1, we also have $B_1^K(D) = B_2^K(D)$. Therefore, $B_5^K(D) = B_1^K(D)$. □

Theorem 3

Proof. We know that $B_3^K(D) \geq B_4^K(D)$ holds automatically because $\Omega_4^K \subseteq \Omega_3^K$. We proceed to prove that $B_3^K(D) \leq B_4^K(D)$. Let us consider an arbitrary feasible solution $(\zeta, \gamma, \tilde{\gamma}, \phi, \varepsilon) \in \Omega_3$. We discuss two cases.

- Case 1. If ϕ contains no empty clauses, then it is valid as an instance of Ω_4 .
- Case 2. If ϕ contains empty clauses, there exist a pair (t', c') such that $\overline{\phi}_{f,c'}^{t'} = \underline{\phi}_{f,c'}^{t'} = 0$, for all $f \in \mathcal{F}$. By constraints (4b)-(4c), $g_{i,t',c'}(\phi) = \gamma_{i,t',c'} = 1$ for all $i \in \mathcal{I}$, so that constraints (4f) implies $1 = \gamma_{i,t',c'} \leq \tilde{\gamma}_{i,c'} = 1$, for all i . Hence, for all the other $c \in \mathcal{C} \setminus \{c'\}$, constraints Γ^{CC} in (4e) entails $\tilde{\gamma}_{i,c} \leq \sum_{t \in \mathcal{T}} \gamma_{i,t,c} = 0$. The corresponding objective value of this solution of problem $(3)B_{M=1}^K(D)$ is obtained by using the constraints on ζ_i , which establish that $\zeta_i \leq \sum_{t \in \mathcal{T}} \gamma_{i,t,c}$, for all $i : u_{i,c} = 1, c \in \mathcal{C}$. Therefore,

$$\sum_{i \in \mathcal{I}} \zeta_i = \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{I}} u_{i,c} \max_t \{g_{i,t,c}(\phi)\} = \sum_{i \in \mathcal{I}} \sum_{c \in \mathcal{C}} \mathbf{1}\{c = c'\}.$$

In fact, for each $i \in \mathcal{I}$, either $\zeta_i \leq |\mathcal{T}|$ if $u_{i,c'} = 1$ or $\zeta_i \leq 0$ if $u_{i,c'} = 0$. Thus, if ϕ contains empty clauses, the optimal objective value of problem $(3)B_{M=3}^K$ is the number of instance in class c' .

As the DNF needs to classify false for all classes other than c' , we can discuss in further two subcases. This depends on the property of dataset :

- Case 2a. Specify that this depends on the property of dataset (cf. example from print from last meeting) that still validates the proof and is still general. There exists some conjunction that is unsatisfied by all instances v in the data. Namely, there exists some $\overline{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t$ such that $g_{i,t,c}(\phi') = 0$ for all i . In this case, we can construct an instance of problem $(3)B_{M=4}^K$ with the same objective value. Since we want all individuals i to be true in c' and to be false in c , we select two clauses as negation to each other in class c' . This makes the disjunction over that class always true. Next, we make all clauses in all other classes $c \in \mathcal{C} \setminus \{c'\}$, the same as ϕ' which is unsatisfied by all instances $i \in \mathcal{I}$, making the disjunction elsewhere always false. Formally, this solution of problem $(3)B_{M=4}^K$ is explicitly encoded by the following decision variable: let ϕ^* be such that $\overline{\phi}_{c,1}^{*t} = \underline{\phi}_{c,1}^{*t} = 1$ for all $t \in \mathcal{T}$ and for all $c \in \mathcal{C} \setminus \{c'\}$. As for all $c \in \mathcal{C} \setminus \{c'\}$, clauses are encoded as $\overline{\phi}_{f,c}^{*t} = \overline{\phi}_{f,c}^t$ and $\underline{\phi}_{f,c}^{*t} = \underline{\phi}_{f,c}^t$. Notably, by construction $(\zeta, g(\phi^*), \phi^*)$ is an instance of Ω_2 (as there is no empty clause in it) and it has the same objective value as $(\zeta, \gamma, \cdot, \phi, \cdot) \in \Omega_2$.

- Case 2b. There doesn't exist a conjunction the is unsatisfied by all instances x , then the original

problem is not feasible. This case doesn't exist.

Therefore, under all possible cases, any feasible solution in Ω_4 corresponds to a feasible solution in Ω_3 with the same objective value. We conclude that $B_2^K \geq B_3^K$ $\square$

Proposition 1

Proof. To prove this proposition we construct a feasible solution of the continuous relaxation of problem (4) for any specification Ω_M^K for $M = 1, 2, 3, 4, 5$ and $K = \text{OC}, \text{CC}$ as follows:

$$\left\{ \begin{array}{lll} \zeta_i^* & = 1 & i \in \mathcal{I} \\ \gamma_{i,t,c}^* & = \frac{1}{T} & i \in \mathcal{I}, c = u_i \\ \tilde{\gamma}_{i,c}^* & = 1 & i \in \mathcal{I}, c = u_i \\ \gamma_{i,t,c}^* = \tilde{\gamma}_{i,c}^* & = 0 & i \in \mathcal{I}, c \in \mathcal{C} \setminus \{u_i\} \\ \bar{\phi}_{f,c}^{t,*} = \underline{\phi}_{f,c}^{t,*} & = \frac{1}{F} & f \in \mathcal{F}, c \in \mathcal{C}, t \in \mathcal{T} \\ \varepsilon_{t,c} & = 0 & c \in \mathcal{C}, t \in \mathcal{T}. \end{array} \right. \quad (\text{B.1})$$

Using the holding assumption $\frac{1}{T} \leq 1 - \frac{1}{F}$ and $2 \leq D$, we deduce that $\sum_{i \in \mathcal{I}} \zeta_i^* = n$ is optimal, by proving the feasibility of this solution (i.e., it satisfies all constraints). Firstly, the right term of constraints (4b) reduce to $1 - \frac{1}{F}$, which hold if $\frac{1}{T} \leq 1 - \frac{1}{F}$. Further, the right term of constraints (4c) reduce to 0, which always hold. Next, constraints (4d) simplify to $2 \leq D$. Finally, Constraints (4e) and (4f) both hold, as well as constraints (4g) (as both sides of the inequality reduce to 1 under solution (B.1)).

We now show that this solution validates the constraints (4h) for each formulation Ω_M^K . Regarding Ω_1 till Ω_4 , the specific constraints of each formulation are satisfied by the construction. In particular, solution (B.1) satisfies constraints (7) and (8) for preventing empty and contradictory clauses respectively. For Γ^{OC} in (5), the construction satisfies all the constraints (as both side of the inequality constraints reduce to 1). For Γ^{CC} in (5), the construction satisfies all the constraints.

Regarding constraints involving ε , the right term of (9a) in Ω_5 stays positive, therefore the inequality is satisfied. Constraints (9b) establish that $\varepsilon_{t,c}$ must be greater than $\gamma_{i,t,c} - \sum_{f \in F} (\bar{\phi}_{f,c}^t + \underline{\phi}_{f,c}^t) \leq \frac{1}{T} - \sum_{f \in F} \frac{2}{F} < 0$, which always holds. Since $\varepsilon_{t,c} = 0$, constraints (9c) hold. $\square$

Theorem 4

Proof. Let t be a clause for class c such that $\sum_{f \in \mathcal{F}} \phi_{f,c}^t \leq D$. Define the set of selected features

$$S := \{f \in \mathcal{F} : \phi_{f,c}^t = 1\}, \quad k := |S| = \sum_{f \in \mathcal{F}} \phi_{f,c}^t \leq D.$$

Choose any subset $R \subseteq \mathcal{F} \setminus S$ with $|R| = D - k$. (If $k = D$, take $R = \emptyset$.) For each assignment $a \in \{0, 1\}^R$, define a D -saturation of t by $(\bar{\phi}_{\cdot,c}^{t,a}, \underline{\phi}_{\cdot,c}^{t,a})$ where, for each $f \in \mathcal{F}$,

$$(\bar{\phi}_{f,c}^{t,a}, \underline{\phi}_{f,c}^{t,a}) = \begin{cases} (\bar{\phi}_{f,c}^t, \underline{\phi}_{f,c}^t) & \text{if } f \in S, \\ (a_f, 1 - a_f) & \text{if } f \in R, \\ (0, 0) & \text{if } f \in \mathcal{F} \setminus (S \cup R). \end{cases}$$

Let $\mathcal{A}(t) := \{0, 1\}^R$. By construction, each saturation adds exactly one literal for each $r \in R$, hence

$$\sum_{f \in \mathcal{F}} \phi_{f,c}^{t,a} = |S| + |R| = k + (D - k) = D,$$

so every $a \in \mathcal{A}(t)$ is D -saturated (namely, $\sum_{f \in \mathcal{F}} \phi_{f,c}^{t,a} = D$). We now show that, for every $\mathbf{v} \in \{0, 1\}^F$,

$$f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) = \bigvee_{a \in \mathcal{A}(t)} f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^{t,a}, \underline{\phi}_{\cdot,c}^{t,a}). \quad (\text{B.2})$$

Fix $\mathbf{v} \in \{0, 1\}^F$. Since saturations leave literals in S unchanged and append one literal for each $r \in R$, we have

$$f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^{t,a}, \underline{\phi}_{\cdot,c}^{t,a}) = f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) \cdot \prod_{r \in R} (v_r^{a_r} (1 - v_r)^{1-a_r}). \quad (\text{B.3})$$

Consider the disjunction over all $a \in \{0, 1\}^R$. In Boolean algebra this corresponds to

$$\bigvee_{a \in \{0,1\}^R} \prod_{r \in R} (v_r^{a_r} (1 - v_r)^{1-a_r}).$$

For the particular assignment $a^* \in \{0, 1\}^R$ defined by $a_r^* = v_r$ for each $r \in R$, every factor satisfies $v_r^{a_r^*} (1 - v_r)^{1-a_r^*} = 1$, so the corresponding product equals 1. Hence, the disjunction over all a is:

$$\bigvee_{a \in \{0,1\}^R} \prod_{r \in R} (v_r^{a_r} (1 - v_r)^{1-a_r}) = 1. \quad (\text{B.4})$$

Using (B.3) and distributing $\wedge$ over $\vee$ (equivalently, factoring out the common term), we obtain

$$\bigvee_{a \in \mathcal{A}(t)} f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^{t,a}, \underline{\phi}_{\cdot,c}^{t,a}) = f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t) \cdot \bigvee_{a \in \{0,1\}^R} \prod_{r \in R} (v_r^{a_r} (1 - v_r)^{1-a_r}) = f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^t, \underline{\phi}_{\cdot,c}^t),$$

where the last equality follows from (B.4). Therefore, for any clause t with at most D literals, there exists a finite set of clauses $\mathcal{A}(t)$ such that:

(i) every $a \in \mathcal{A}(t)$ is D -saturated in the sense that $\sum_{f \in \mathcal{F}} \phi_{f,c}^{t,a} = D$;

(ii) for every $\mathbf{v} \in \{0, 1\}^F$,

$$f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^t, \phi_{\cdot,c}^t) = \bigvee_{a \in \mathcal{A}(t)} f_{\mathbf{v}}(\bar{\phi}_{\cdot,c}^{t,a}, \phi_{\cdot,c}^{t,a}). \quad (\text{B.5})$$

Finally, apply (B.5) to every clause t appearing in the disjunction $\bigvee_{t \in \mathcal{T}}(\cdot)$ of (1): replacing a disjunct by a logically equivalent disjunction preserves the Boolean value of the overall DNF for every $\mathbf{v}$, and all resulting clauses are D -saturated. $\square$

Corollary 1

Proof. Assume $D \leq F$. Let $Z_{\leq D}$ denote the optimal value of problem (4) when clauses are allowed to contain at most D literals. Let $Z_{=D}$ denote the optimal value of the saturated formulation in which, for each class, at most $T_{[=D]}$ active non-false clauses are used and every active clause contains exactly D literals. Since every exact- D clause is, in particular, a clause with at most D literals, the feasible classifiers of the saturated formulation are feasible for the original at-most- D formulation. Hence $Z_{=D} \leq Z_{\leq D}$.

We prove the reverse inequality. Let Φ be an optimal solution of problem (4) in the at-most- D formulation. For each class $c \in \mathcal{C}$, define the truth value induced by Φ on observation i by

$$h_{i,c}(\Phi) = \max_{t \in \mathcal{T}} f_{v_i}(\phi_{\cdot,c}^t, \bar{\phi}_{\cdot,c}^t).$$

The objective value and the categorical or ordinal feasibility conditions of problem (4) depend on Φ only through the collection $\{h_{i,c}(\Phi) : i \in \mathcal{I}, c \in \mathcal{C}\}$. Indeed, in the categorical case, feasibility and correctness are determined by the class truth values $h_{i,c}(\Phi)$. In the ordinal case, the first-true rule is also completely determined by these same truth values.

Consider any clause of Φ . If the clause is contradictory, then it is identically false and may be removed without changing any truth value. If the clause is non-false and contains $k \leq D$ literals, then, by Theorem 4, it can be replaced by a finite disjunction of non-false clauses containing exactly D literals, without changing its Boolean value on any $\mathbf{v} \in \{0, 1\}^F$. Applying this replacement to every clause of every class yields a saturated DNF, denoted by $\hat{\Phi}$, such that

$$h_{i,c}(\hat{\Phi}) = h_{i,c}(\Phi), \quad \forall i \in \mathcal{I}, \forall c \in \mathcal{C}.$$

Therefore $\hat{\Phi}$ has the same objective value as Φ and satisfies the same categorical or ordinal classification

conditions on the training sample. It remains to show that at most

$$T_{[=D]} = \min \left\{ I, \binom{F}{D} 2^D \right\}$$

active clause slots per class are sufficient. Fix a class c , and let $\mathcal{Q}_c$ be the set of distinct exact- D non-false clauses appearing in the saturated DNF for class c . An exact- D non-false clause is obtained by choosing D distinct features and one polarity for each chosen feature. Hence

$$|\mathcal{Q}_c| \leq \binom{F}{D} 2^D.$$

For each clause $q \in \mathcal{Q}_c$, define its support on the training sample as $S(q) = \{i \in \mathcal{I} : f_{v_i}(q) = 1\}$. Let

$$S_c = \bigcup_{q \in \mathcal{Q}_c} S(q)$$

be the set of training observations for which the class- c DNF is true. For every $i \in S_c$, choose one clause $q_i \in \mathcal{Q}_c$ such that $i \in S(q_i)$, and define $\mathcal{Q}'_c = \{q_i : i \in S_c\}$. If $S_c = \emptyset$, we take $\mathcal{Q}'_c = \emptyset$, so that the class- c DNF is false on the training sample. Otherwise, $|\mathcal{Q}'_c| \leq |S_c| \leq I$. Moreover, since $\mathcal{Q}'_c \subseteq \mathcal{Q}_c$, we have

$$\bigcup_{q \in \mathcal{Q}'_c} S(q) \subseteq S_c.$$

The reverse inclusion also holds: for every $i \in S_c$, the clause $q_i \in \mathcal{Q}'_c$ was chosen so that $i \in S(q_i)$. Therefore

$$\bigcup_{q \in \mathcal{Q}'_c} S(q) = S_c.$$

Thus replacing $\mathcal{Q}_c$ by $\mathcal{Q}'_c$ preserves the truth value of the class- c DNF on every training observation. Since $\mathcal{Q}'_c \subseteq \mathcal{Q}_c$, we also have

$$|\mathcal{Q}'_c| \leq |\mathcal{Q}_c| \leq \binom{F}{D} 2^D.$$

Combining the two bounds gives

$$|\mathcal{Q}'_c| \leq \min \left\{ I, \binom{F}{D} 2^D \right\} = T_{[=D]}.$$

Applying this reduction independently to every class produces a saturated DNF using at most $T_{[=D]}$ active exact- D clauses per class and satisfying

$$\sum_{f \in \mathcal{F}} (\phi_{f,c}^t + \bar{\phi}_{f,c}^t) = D$$

for every active clause t of every class c . This reduced saturated DNF has the same values $h_{i,c}$ as Φ on the training sample, and therefore has the same feasibility status and the same objective value in problem (4). Hence, $Z_{\leq D} \leq Z_{=D}$. Together with $Z_{=D} \leq Z_{\leq D}$, this gives $Z_{\leq D} = Z_{=D}$. Therefore, the saturated DNF formulation can be expressed, without loss of optimality in problem (4), using at most $T_{[=D]}$ clause slots per class and imposing the exact- D cardinality constraint on each active clause. $\square$

Lemma 1

Proof. For each (t, c) , the associated clause has exactly D literals. Since $D < F$, there exists at least one feature not used in that clause. Let $f_{t,c}$ be the first feature not appearing in that clause. For each (t, c) , where t varies from 1 up to T , and $c \in \mathcal{C}$, we construct two new clauses indexed by (t, c) and $(T + t, c)$ as follows:

$$\begin{aligned} \overline{\phi}'_{f,c}{}^t &= \begin{cases} 1 & \text{if } f = f_{t,c}, \\ \overline{\phi}_{f,c}{}^t & \text{otherwise,} \end{cases} & \underline{\phi}'_{f,c}{}^t &= \begin{cases} 0 & \text{if } f = f_{t,c}, \\ \underline{\phi}_{f,c}{}^t & \text{otherwise,} \end{cases} \\ \overline{\phi}'_{f,c}{}^{T+t} &= \begin{cases} 0 & \text{if } f = f_{t,c}, \\ \overline{\phi}_{f,c}{}^t & \text{otherwise,} \end{cases} & \underline{\phi}'_{f,c}{}^{T+t} &= \begin{cases} 1 & \text{if } f = f_{t,c}, \\ \underline{\phi}_{f,c}{}^t & \text{otherwise.} \end{cases} \end{aligned}$$

Each original clause is thus replaced by two clauses: one where the new literal appears positively, and one where it appears negatively. Therefore, each new clause contains exactly $D + 1$ literals and contains $2T$ clauses. Since the disjunction includes both cases for $f_{t,c}$, the overall logical function remains unchanged. Hence, the two DNFs yield the same evaluation, so they are equivalent in terms of the Boolean evaluation. $\square$

C MILP Decision Tree Formulations

This appendix presents two MILP formulations for DTs. First, we present the **OCT** formulation of Bertsimas and Dunn (2017), followed by the **FlowOCT** formulation of Aghaei et al. (2025).

OCT formulation (Bertsimas and Dunn 2017) Let $\mathcal{N}$ and $\mathcal{L}$ denote the sets of branching nodes and leaves, respectively. For each leaf $\ell \in \mathcal{L}$, let $\mathcal{A}_L(\ell)$ and $\mathcal{A}_R(\ell)$ denote the sets of ancestor nodes where the left and right branch, respectively, must be followed to reach ℓ . We use the convention that the left branch corresponds to feature value zero and the right branch corresponds to feature value one.

For a given observation i and a class c , we use the label indicator $u_{i,c} \in \{0, 1\}$ introduced in Section 2. Let $b_{n,f}$ indicate whether feature f is selected for splitting at branching node $n \in \mathcal{N}$, let $w_{\ell,c}$ denote whether class c is assigned to leaf $\ell \in \mathcal{L}$, and let $z_{i\ell}$ track whether observation i reaches leaf ℓ . Together, these variables encode the tree structure, observation assignment, and leaf-class prediction within the mixed-integer program. A compact version of the **OCT** formulation, adapted to binary features and to the notation of this paper, is given by:

$$\left\{ \begin{array}{ll} \min & \sum_{\ell \in \mathcal{L}} L_{\ell} \quad (C.1a) \\ \text{s. to} & L_{\ell} \geq \sum_{i \in \mathcal{I}} z_{i\ell}(1 - u_{i,c}) - |\mathcal{I}|(1 - w_{\ell,c}), \quad \forall c \in \mathcal{C}, \ell \in \mathcal{L} \quad (C.1b) \\ & \sum_{c \in \mathcal{C}} w_{\ell,c} = 1, \quad \forall \ell \in \mathcal{L} \quad (C.1c) \\ & \sum_{f \in \mathcal{F}} b_{n,f} v_{i,f} \geq \frac{1}{2} - (1 - z_{i\ell}), \quad \forall i \in \mathcal{I}, \ell \in \mathcal{L}, n \in \mathcal{A}_R(\ell) \quad (C.1d) \\ & \sum_{f \in \mathcal{F}} b_{n,f} v_{i,f} \leq \frac{1}{2} + (1 - z_{i\ell}), \quad \forall i \in \mathcal{I}, \ell \in \mathcal{L}, n \in \mathcal{A}_L(\ell) \quad (C.1e) \\ & \sum_{\ell \in \mathcal{L}} z_{i\ell} = 1, \quad \forall i \in \mathcal{I} \quad (C.1f) \\ & \sum_{f \in \mathcal{F}} b_{n,f} = 1, \quad \forall n \in \mathcal{N} \quad (C.1g) \\ & L_{\ell} \geq 0, \quad \forall \ell \in \mathcal{L} \quad (C.1h) \\ & z_{i\ell} \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \ell \in \mathcal{L} \quad (C.1i) \\ & b_{n,f} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, f \in \mathcal{F} \quad (C.1j) \\ & w_{\ell,c} \in \{0, 1\}, \quad \forall \ell \in \mathcal{L}, c \in \mathcal{C}. \quad (C.1k) \end{array} \right.$$

Constraint (C.1b) ensures that L_ℓ correctly counts the number of misclassified observations assigned to leaf ℓ . Indeed, if leaf ℓ is assigned to class c , i.e., $w_{\ell,c} = 1$, then (C.1b) reduces to

$$L_\ell \geq \sum_{i \in \mathcal{I}} z_{i\ell}(1 - u_{i,c}),$$

which is the number of observations routed to ℓ whose true class is not c . Constraint (C.1c) enforces that each leaf predicts exactly one class. Constraints (C.1d)–(C.1e) ensure that an observation can reach a leaf only if its features satisfy all splitting decisions along the corresponding root-to-leaf path. Constraint (C.1f) guarantees that each observation is assigned to exactly one leaf, while (C.1g) ensures that each branching node selects a single feature for splitting. Together, these constraints encode the tree logic within the mixed-integer program, and the objective (C.1a) minimizes the total misclassification loss.

FlowOCT formulation (Aghaei et al. 2025) In the DT max-flow formulation presented below, the tree is augmented by a single sink node connected to all nodes in $\mathcal{N} \cup \mathcal{L}$. With a slightly different choice of notation with respect to Aghaei et al. (2025), nodes are assigned numerical labels in breadth-first order. For any internal node $n \in \mathcal{N}$, its left descendant is encoded by $2n$ and its right descendant by $2n + 1$. This implies that the parent of a node $n > 1$ is $\lfloor n/2 \rfloor$, and the tree layer where n is located is $\lambda(n) = 1 + \lfloor \log_2(n) \rfloor$ when the root is indexed as layer one.

We define four groups of decision variables:

- $b_{n,f}$: the binary indicator of whether the tree branches on feature $f \in \mathcal{F}$ at node $n \in \mathcal{N}$;
- $w_{n,c}$: the binary indicator of whether node $n \in \mathcal{N} \cup \mathcal{L}$ predicts class $c \in \mathcal{C}$;
- $z_{n,i}$: the indicator of whether the flow associated with observation i reaches node $n \in \mathcal{N} \cup \mathcal{L}$;
- $r_{n,i}$: the indicator of whether observation i is correctly classified and sent to the sink node from $n \in \mathcal{N} \cup \mathcal{L}$.

The mixed-integer linear programming formulation from Aghaei et al. (2025), adapted to the

present notation, can be written as follows:

$$\left\{ \begin{array}{ll} \max & \sum_{i \in \mathcal{I}} \sum_{n \in \mathcal{N} \cup \mathcal{L}} r_{n,i} \quad (C.2a) \\ \text{subject to} & z_{1,i} = 1 \quad \forall i \in \mathcal{I} \quad (C.2b) \\ & \sum_{f \in \mathcal{F}} b_{n,f} + \sum_{c \in \mathcal{C}} w_{n,c} = 1 \quad \forall n \in \mathcal{N} \quad (C.2c) \\ & \sum_{c \in \mathcal{C}} w_{n,c} = 1 \quad \forall n \in \mathcal{L} \quad (C.2d) \\ & z_{2n,i} + z_{2n+1,i} + r_{n,i} \leq z_{n,i} \quad \forall n \in \mathcal{N}, i \in \mathcal{I} \quad (C.2e) \\ & r_{n,i} \leq z_{n,i} \quad \forall n \in \mathcal{L}, i \in \mathcal{I} \quad (C.2f) \\ & z_{2n,i} \leq \sum_{f \in \mathcal{F}} (1 - v_{i,f}) b_{n,f} \quad \forall n \in \mathcal{N}, i \in \mathcal{I} \quad (C.2g) \\ & z_{2n+1,i} \leq \sum_{f \in \mathcal{F}} v_{i,f} b_{n,f} \quad \forall n \in \mathcal{N}, i \in \mathcal{I} \quad (C.2h) \\ & u_{i,c} r_{n,i} \leq w_{n,c} \quad \forall n \in \mathcal{N} \cup \mathcal{L}, i \in \mathcal{I}, c \in \mathcal{C} \quad (C.2i) \\ & b_{n,f} \in \{0, 1\} \quad \forall n \in \mathcal{N}, f \in \mathcal{F} \quad (C.2j) \\ & w_{n,c} \in \{0, 1\} \quad \forall n \in \mathcal{N} \cup \mathcal{L}, c \in \mathcal{C} \quad (C.2k) \\ & z_{n,i}, r_{n,i} \in \{0, 1\} \quad \forall n \in \mathcal{N} \cup \mathcal{L}, i \in \mathcal{I}. \quad (C.2l) \end{array} \right.$$

Constraint (C.2b) initializes one unit of observation-specific flow at the root node. Constraints (C.2c) ensure that each internal node either selects one feature for branching or is pruned and assigned to one class, but not both. Constraints (C.2d) assign exactly one class to each leaf. Constraints (C.2e) are flow-conservation constraints with possible blockage: if observation i reaches an internal node n , then at most one unit of flow can move to the left child, move to the right child, or be sent directly to the sink from node n if that node is used as a terminal prediction node.¹⁰ Similarly, constraints (C.2f) establish that flow can be sent to the sink from a leaf node only if observation i reaches that leaf. Constraints (C.2g) and (C.2h) impose consistency between the selected split and the observed feature vector: an observation can move to the left child of node n only if the selected feature has value zero, and it can move to the right child only if the selected feature has value one. Constraint (C.2i) ensures that observations routed to the sink are correctly classified: if $r_{n,i} = 1$, then node n must be assigned to the true class of observation i . The objective (C.2a) maximizes the number of

¹⁰Aghaei et al. (2025) interpret these constraints as flow-conservation constraints, where a data point arriving at a node may leave that node through one of its descendants or through the sink, while unassigned flow is blocked and therefore not counted as correctly classified.

correctly classified observations.

D Quantum Annealing

Quantum annealing (QA) can be regarded as a continuous-time analogue of simulated annealing (SA): both search a binary space associated with a problem such as (11), but QA does so through a dynamics that evolves continuously in time.

Boolean optimization problems, such as (11), were among the first applications of SA (Kirkpatrick et al. 1983). More recently, Bouhmala (2019) employed a hybrid SA algorithm for the MAX-SAT problem, confirming the applicability of this technique to hard Boolean optimization problems.

Of particular interest and relevance to this paper is the work of Heath et al. (1993), who focused specifically on the optimal characterization of univariate classification trees.

When SA is used to tackle the constrained version of problem (11), it takes the form of a discrete-time Markov chain that moves through the feasible region $\mathcal{X}$ delimited by the corresponding constraints; its transition probabilities are ideally tuned so that the chain’s stationary distribution concentrates on an optimal solution. In the QUBO reformulation used in this paper, the constraints are instead incorporated through penalty terms, so that the search takes place over a binary hypercube. QA acts on this penalized formulation by evolving a quantum state whose measurement induces a probability distribution over binary vectors. More formally, Kadowaki and Nishimori (1998) and Albash and Lidar (2018) cast QA as an adiabatic procedure for minimizing a quadratic form $\mathbf{y}^\top \mathbf{J} \mathbf{y}$ in binary variables $\mathbf{y} \in \{0, 1\}^{\tilde{n}}$ (i.e., a QUBO), which makes it directly applicable to unconstrained penalty reformulations of (11). In our pipeline, QA is therefore used as an incumbent-generation layer for the DNF formulations, rather than as a replacement for exact MILP optimization.

Starting from this penalty reformulation, QA aims to generate low-energy binary vectors in $\{0, 1\}^{\tilde{n}}$,¹¹ where, in a compact notation,

$$\tilde{n} = |\mathcal{I}| |\mathcal{T}| |\mathcal{C}| + |\mathcal{I}| + 2|\mathcal{F}| |\mathcal{T}| |\mathcal{C}| + S,$$

and S collects the ancillary and formulation-specific binary variables, including slack encodings and activation or emptiness variables when they are retained. The construction relies on the state vector $\Psi(t) \in \mathbb{C}^{2^{\tilde{n}}}$, evolving in continuous time according to the Schrödinger equation

$$i\hbar \frac{\partial \Psi(t)}{\partial t} = H(t) \Psi(t), \tag{D.1}$$

where i is the imaginary unit, $\hbar$ is the reduced Planck constant, and $H(t)$ is a time-dependent

¹¹Every feasible vector of the constrained binary formulation can be extended by suitable slack variables to an element of $\{0, 1\}^{\tilde{n}}$; infeasible assignments remain in the QUBO search space but are penalized.

Hamiltonian operator.¹² A standard representation is

$$H(t) = A(t)H_B + B(t)H_P,$$

where $A(t)$ decreases during the anneal, $B(t)$ increases, H_B is the initial mixing Hamiltonian, and H_P is the problem Hamiltonian encoding the QUBO objective. Using the notation

$$\mathbb{I}_{\otimes}^m = I_2 \otimes I_2 \otimes \cdots \otimes I_2 \in \mathbb{R}^{2^m \times 2^m},$$

for the tensor product of m copies of the two-dimensional identity matrix I_2 , one may write

$$H_B = - \sum_{i=1}^{\tilde{n}} \mathbb{I}_{\otimes}^{i-1} \otimes \sigma^x \otimes \mathbb{I}_{\otimes}^{\tilde{n}-i}, \quad \sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The QUBO objective is encoded through the diagonal operators

$$\hat{Y}_i = \mathbb{I}_{\otimes}^{i-1} \otimes \frac{I_2 - \sigma^z}{2} \otimes \mathbb{I}_{\otimes}^{\tilde{n}-i}, \quad \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

so that, up to additive constants,

$$H_P = \sum_{i=1}^{\tilde{n}} J_{ii} \hat{Y}_i + \sum_{i=1}^{\tilde{n}-1} \sum_{j=i+1}^{\tilde{n}} (J_{ij} + J_{ji}) \hat{Y}_i \hat{Y}_j.$$

At the end of the anneal, measuring $\Psi(t)$ in the computational basis produces a binary vector $\mathbf{y}$ with probability $|\langle \mathbf{y} | \Psi(t) \rangle|^2$. In the ideal adiabatic limit, this distribution concentrates on ground states of H_P , which correspond to minimizers of the QUBO energy.

The appeal of representing the solution of (11) through the process $\{|\Psi(t)|^2 : t \geq 0\}$ stems from quantum superposition and from the physical implementation of the annealing dynamics. Classical simulation approaches must explicitly generate transitions or samples to approximate a limiting distribution. QA instead evolves a superposition of computational-basis states under (D.1) and produces samples only upon measurement. Repeated reads are still required to estimate the output distribution, but the annealing trajectory itself is implemented directly by the analog device.

The QA computational experiments in this work are performed using D-Wave's¹³ hybrid solver,

¹²The behavior of this time-dependent Hamiltonian rests on the adiabatic principle: if the Hamiltonian is varied sufficiently slowly and the system is initialized in a ground state, then the system remains close to an instantaneous ground state throughout the evolution (Kadowaki and Nishimori 1998, Albash and Lidar 2018).

¹³D-Wave is a Canadian quantum computing company that provides commercially available quantum annealers based on superconducting flux qubits implemented with Josephson junctions.

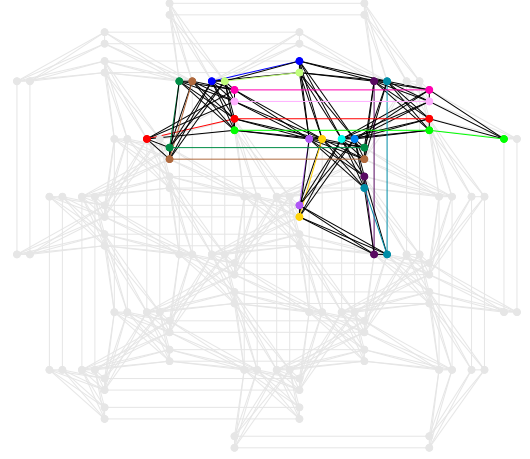
with access to the *Advantage 6.4* quantum processing unit (QPU). This QPU incorporates 5,616 operational qubits arranged in the fixed *Pegasus* topology, where each qubit is connected to up to 15 neighbors. In total, the architecture comprises roughly 40,088 active couplers, implemented as microscopic niobium loops interacting through Josephson junction networks and governed by high-precision analog control systems.

To run a QUBO on physical annealing hardware, each decision variable must be mapped onto the Pegasus graph through a process known as *embedding*. In this mapping, graph vertices represent qubit locations on the chip, and edges represent the available couplings, ensuring that quadratic terms in the QUBO correspond to physically adjacent qubits (Serra et al. 2022). The qubits themselves are superconducting devices operating at millikelvin temperatures, and the resulting hardware behaves as an analog computer implementing the continuous-time evolution that induces the measurement probabilities $\{|\Psi(t)|^2 : t \geq 0\}$.

Embedding adapts the QUBO to the QPU’s limited connectivity by translating it into an equivalent Ising model and representing one logical variable by a chain of physically connected qubits. An example of such a process is demonstrated in Figure 5, where a QUBO problem is embedded in the fixed Pegasus architecture of the QPU. Because Pegasus is not fully connected, additional qubits and chains are often required. Finding an optimal embedding, one that minimizes the number of extra qubits and chain lengths, is itself NP-hard (Bernal et al. 2020), so D-Wave provides efficient heuristics that usually yield high-quality embeddings when sufficient hardware resources are available. Qubits within a chain are forced to take the same value through penalty terms added to the objective. For instance, in Figure 6, subfigure (a) shows no direct coupler between x_2 and x_3 . An auxiliary qubit y_4 , connected to both, is introduced and chained to x_3 using a penalty such as $\lambda_c(y_4 - x_3)^2 = \lambda_c(y_4 + x_3 - 2y_4x_3)$, which encourages y_4 to match x_3 , thereby creating the required virtual connectivity, as shown in subfigure (b). Note that the embedding displayed in subfigure (c), with additional qubits y_4 and y_5 , is also valid, but is less resource-efficient and more susceptible to errors than the former, making shorter-chain embeddings generally preferable. The parameter λ_c , known as the *chain strength*, must be chosen sufficiently large to preserve chain consistency throughout the annealing process.

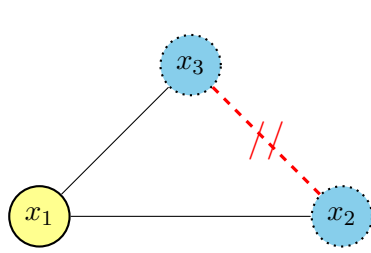
q1	-11448	1760	4400	880	3520	2640	1760	4400	2640	880	880	1760	3520	7040
q2	0	-6163	2200	440	1760	1320	880	2200	1320	440	440	880	1760	3520
q3	0	0	-13757	1100	4400	3300	2200	5500	3300	1100	1100	2200	4400	8800
q4	0	0	0	-3192	880	660	440	1100	660	220	220	440	880	1760
q5	0	0	0	0	-11449	2640	1760	4400	2640	880	880	1760	3520	7040
q6	0	0	0	0	0	-8916	1320	3300	1980	660	660	1320	2640	5280
q7	0	0	0	0	0	0	-6164	2200	1320	440	440	880	1760	3520
q8	0	0	0	0	0	0	0	-13760	3300	1100	1100	2200	4400	8800
q9	0	0	0	0	0	0	0	0	-8915	660	660	1320	2640	5280
q10	0	0	0	0	0	0	0	0	0	-3191	220	440	880	1760
q11	0	0	0	0	0	0	0	0	0	0	-3190	440	880	1760
q12	0	0	0	0	0	0	0	0	0	0	0	-6160	1760	3520
q13	0	0	0	0	0	0	0	0	0	0	0	0	-11440	7040
q14	0	0	0	0	0	0	0	0	0	0	0	0	0	-19360
	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14

(a) Original QUBO problem with 14 variables.

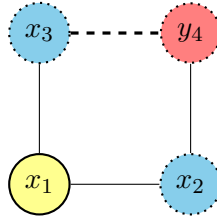


(b) Embedded QUBO problem in the Pegasus topology, via the addition of 18 physical qubits, for a total of 32 physical qubits.

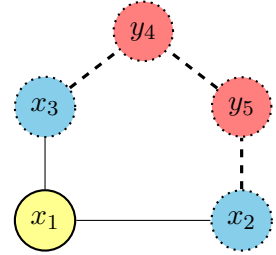
Figure 5: Example of a 14-variable QUBO formulation of a simple 0-1 knapsack problem (10 original variables plus 4 slack variables used for the capacity constraint) and its corresponding embedding in the Pegasus topology. The embedding requires 18 additional physical qubits and results in a maximum chain length of three.



a) Original QUBO graph



b) Embedded QUBO with one auxiliary qubit



c) Embedded QUBO with two auxiliary qubits

Figure 6: Example of embedding a 3-vertex clique onto a qubit graph topology that lacks sufficient connectivity. The left panel depicts the original QUBO/Ising graph, with a red dashed edge indicating that qubits are not physically connected in the QPU. The center panel shows an embedding using one additional qubit (y_4), while the right panel shows a similar but less efficient embedding using two additional qubits (y_4 and y_5). The black dashed lines correspond to ferromagnetic couplings used to form chains.

As a consequence of the embedding process, the number of required physical qubits may substantially exceed the original number of decision variables. This overhead generally increases with the density of the input graph: the more densely connected the logical model, the greater the number of qubits and the longer the chains typically required to represent it. Longer chains are particularly susceptible to the phenomenon known as a “chain break,” which arises when not all qubits in a chain assume the same value after the annealing process, increasing the risk of producing infeasible solutions.

At the current stage of quantum hardware development, limitations in qubit availability, connectivity, and embedding efficiency restrict the size of problem instances that can be practically executed on

the D-Wave annealer. Consequently, for the QUBO formulation of problem (11), only small- to medium-scale datasets can be accommodated, and only for relatively shallow maximum depths D . This hardware bottleneck is precisely why the saturation-based reductions developed in the main text are useful for the QUBO layer. As quantum architectures improve, these constraints are expected to diminish, ultimately enabling the treatment of larger instances and deeper maximum depths.

E Data selection and processing

This appendix describes the data preprocessing for the seven benchmark datasets used in our study, as summarized in Table 2. Since all formulations considered in the paper require binary features, the preprocessing primarily consists of transforming categorical and ordinal attributes into binary representations.

For categorical features, one-hot encoding was applied to create binary dummy variables. When appropriate, one dummy variable per original feature was discarded to avoid redundancy, so that the omitted category can be inferred from the remaining dummies. Ordinal features were encoded through cumulative threshold-based binaries in order to preserve their ordering without using a full one-hot expansion. Labels were mapped to integers, either binary or multi-class depending on the dataset. The differences between our feature counts and those reported in Bertsimas and Dunn (2017) and Aghaei et al. (2025) are due to these alternative encoding choices.

E.1 Monk datasets

All three **monk** datasets have 6 categorical features, represented as X_1 through X_6 . The number of values taken by each feature is summarized in Table E.1.

dataset	I	C	X_1-X_2	X_3	X_4	X_5	X_6	# Dummy Vars
monk1	554	2	3 values each	2 values	3 values	4 values	2 values	11 or 15
monk2	599	2	3 values each	2 values	3 values	4 values	2 values	11 or 15
monk3	552	2	3 values each	2 values	3 values	4 values	2 values	11 or 15

Table E.1: Description of the **monk** datasets. I denotes the number of observations, C is the number of classes, X_1-X_6 indicate feature domains, and # Dummy Vars refers to the number of binary features obtained after encoding.

As reported in Table E.1, the number of binary variables is 11 when one dummy variable is discarded for each original non-binary feature, while it is 15 when all levels of the non-binary features are retained and the two binary features are kept as single binary variables. Our work and Bertsimas and Dunn (2017) use the reduced 11-variable encoding, while Aghaei et al. (2025) use the 15-variable encoding.

E.2 Balance-Scale dataset

The **balance-scale** dataset has 4 categorical features, X_1 through X_4 . The number of values taken by each feature is summarized in Table E.2.

dataset	I	C	X_1	X_2	X_3	X_4	# Dummy Vars
balance-scale	625	3	5 values	5 values	5 values	5 values	16 or 20

Table E.2: Description of the **balance-scale** dataset. I denotes the number of observations and C is the number of classes. Each feature takes integer values from 1 to 5 and is treated as categorical; full one-hot encoding produces 20 dummy variables, while removing one dummy per feature produces 16 binary variables.

As shown in Table E.2, we use the 16 dummy variables obtained after removing the redundant categories. Bertsimas and Dunn (2017) use the 4 original categorical variables directly, while Aghaei et al. (2025) use 20 dummy variables.

E.3 Tic-Tac-Toe dataset

The **tic-tac-toe** dataset, abbreviated as **tic**, contains 958 instances, with 9 original categorical features. Each feature represents a board position and takes values **x** for the X-player, **o** for the O-player, or **b** for blank. The label is binary: **positive** if X wins and **negative** otherwise, mapped to 1 and 0. These 9 features are ternary. Table E.3 summarizes the original features and the corresponding dummy-variable encoding.

dataset	I	C	X_1-X_9	# Dummy Vars
tic	958	2	3 values each	18

Table E.3: Description of the **tic-tac-toe** dataset. I denotes the number of observations and C is the number of classes. Each of the 9 features X_1-X_9 takes 3 categorical values; partial one-hot encoding produces 18 binary variables.

As reported in Table E.3, and consistently with Bertsimas and Dunn (2017) and Aghaei et al. (2025), our encoding of these 9 ternary features comprises 18 dummy variables.

E.4 Car-Evaluation dataset

The **car-evaluation** dataset, abbreviated as **car**, contains 1728 observations and 6 ordinal variables. The number of levels for each variable is summarized in Table E.4.

dataset	I	C	X_1-X_3	X_4-X_6	# Dummy Vars
car	1728	4	4 levels each	3 levels each	15

Table E.4: Description of the **car-evaluation** dataset. I denotes the number of observations and C is the number of classes. Features X_1-X_3 correspond to **buying**, **maint**, and **doors**; features X_4-X_6 correspond to **persons**, **lug_boot**, and **safety**. Cumulative binary threshold encoding produces 15 binary variables.

The six ordinal features in the **car** dataset are **buying**, **maint**, **doors**, **persons**, **lug_boot**, and **safety**, while the class labels are **unacc**, **acc**, **good**, and **vgood**. As indicated in Table E.4, features

were numerically encoded and transformed using cumulative binary thresholds to preserve ordinality, producing 15 binary variables. The class label was mapped to the integer set $\{0, 1, 2, 3\}$.

E.5 kr-kp dataset

The **kr-kp** dataset, also denoted by **krkp**, corresponds to the King-Rook versus King-Pawn chess endgame problem. It contains 3196 instances with a binary label, **won** or **nowin**. The feature set is almost binary: 35 features are binary after treating **dwipd** as a binary attribute, and one feature, **katri**, is ternary with values **n**, **w**, and **b**. The ternary feature is represented by two dummy variables after dropping one redundant category. Table E.5 summarizes the resulting encoding.

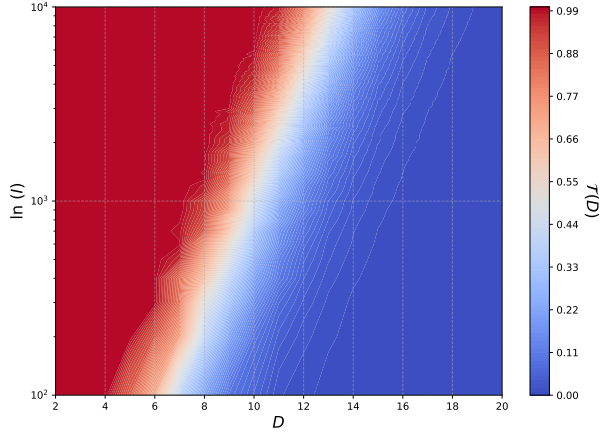
dataset	I	C	35 Binary Features	1 Ternary Feature	# Dummy Vars
kr-kp	3196	2	2 values each	3 values	37

Table E.5: Description of the **kr-kp** dataset. I denotes the number of observations and C is the number of classes. The binary attributes are mapped to 0/1, while the ternary feature **katri** is represented by two dummy variables after removing one redundant category. The final encoding contains 37 binary variables.

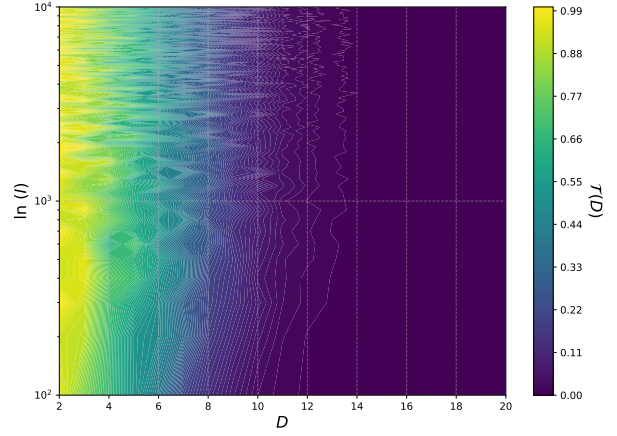
As reported in Table E.5, our preprocessing produces 37 binary variables. Retaining all categories of the ternary feature would instead produce one additional redundant dummy variable, which explains the larger feature count reported in some alternative preprocessing schemes.

E.6 Empirical size of the clause sets induced by CART

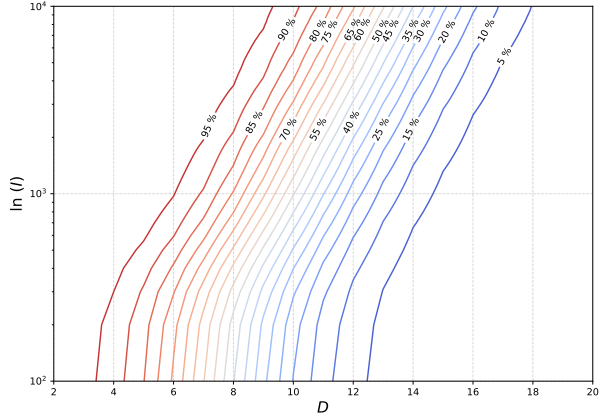
Figure 7 complements the preprocessing description by illustrating the empirical size of the clause sets induced by **CART**. This figure is used to support the saturation and incremental-learning storyline developed in the main text: even though the complete DNF dictionary grows rapidly with the maximum depth D , the number of clauses induced by tree-based warm starts remains much smaller in practice. Panels 7a and 7b report contour area plots for synthetic and real datasets, Panels 7c and 7d show the corresponding contour lines, and Panels 7e and 7f summarize the aggregate behavior.



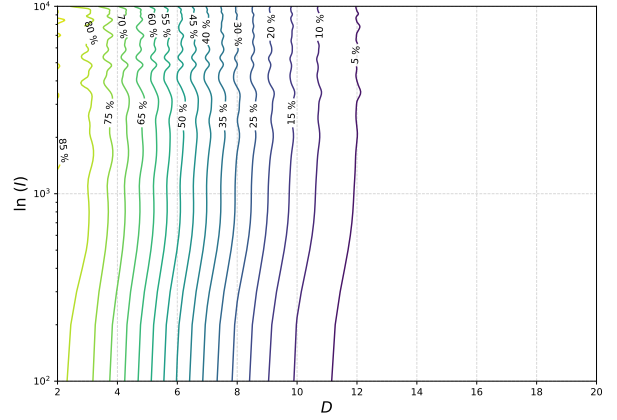
(a) Synthetic datasets: contour area plot showing the degree of saturation using CART's number of conjunctions, depending on number of features and rows.



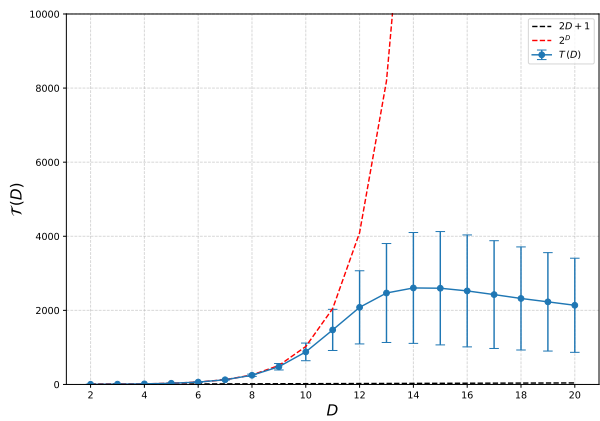
(b) Real datasets: contour area plot showing the degree of saturation using CART's number of conjunctions, depending on number of features and rows.



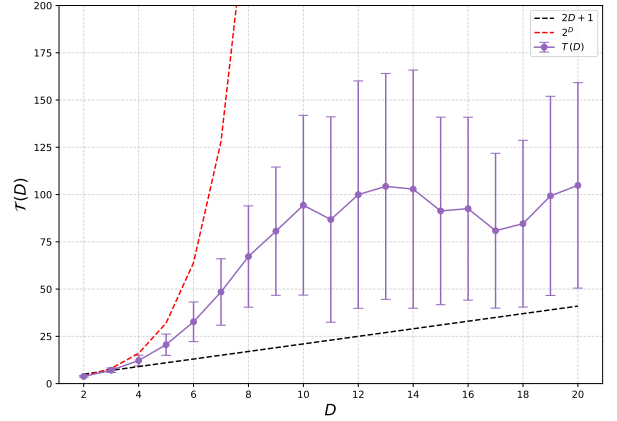
(c) Synthetic datasets: contour line plot showing the top 10 saturation levels.



(d) Real datasets: contour line plot showing the top 10 saturation levels.



(e) Synthetic datasets: aggregate behavior.



(f) Real datasets: aggregate behavior.

Figure 7: Classification tree saturation analysis using CART. Sample-size axes are reported on a logarithmic scale. Subfigures (a) and (b) show contour area plots for synthetic and real datasets, respectively. Subfigures (c) and (d) show the corresponding top 10 contour line levels. Subfigures (e) and (f) display the aggregate behavior for both synthetic and real datasets.

F Numerical results

In this section, we present the detailed numerical results obtained with each of the approaches evaluated in this work.

- Table F.1 presents the results obtained with the *double incremental* approach.
- Table F.2 presents the results obtained with the *single incremental* approach.
- Table F.4 corresponds to the *non-incremental* approach with a pre-determined depth, where the inequality constraint is used.
- Table F.5 reports the same *non-incremental* setting, this time with the equality constraint.
- Table F.6 again considers the *non-incremental* approach with a pre-determined depth and the inequality constraint, now warm-started from the **CART** solution.
- Table F.7 presents the corresponding results for the equality constraint, likewise warm-started from the **CART** solution.

All the aforementioned tables presents the numerical results obtained with their respective algorithmic scheme, comparing the three formulation families Ω_M^{CC} and Ω_M^{OC} across all benchmark datasets. For each dataset–formulation pair, we report the best objective value (**obj**) achieved, the total computational time in seconds (**time**), the final search depth (D^*), and the maximal number of iterations (T^*). These statistics are aggregated over all combinations of D and T for each case $(\Omega_M, \mathbf{data})$, representing the maximal objective value observed, the total solver time, and the mean number of iterations per depth level.

monk1				monk2				monk3				balance-scale				car-evaluation				krkp				tic				
obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	
Ω_1^{CC}	554 (15.24)	4	16	1.00	393 (26.40)	3	9	1.00	546 (35.19)	5	16	1.25	434 (30.01)	3	9	1.00	1694 (652.56)	8	49	1.86	3007 (140.13)	5	16	1.25	957 (580.71)	8	49	1.86
Ω_2^{CC}	554 (17.94)	4	16	1.00	599 (77.81)	6	31	1.20	546 (22.85)	4	16	1.00	434 (30.03)	3	9	1.00	1343 (30.03)	3	9	1.00	3007 (140.15)	5	16	1.25	957 (480.82)	8	43	1.57
Ω_3^{CC}	554 (21.53)	4	16	1.00	599 (80.96)	6	31	1.20	546 (32.66)	6	19	1.40	434 (30.01)	3	9	1.00	1686 (570.61)	8	33	1.57	3007 (140.18)	5	16	1.25	958 (528.11)	8	57	1.86
Ω_4^{CC}	554 (25.39)	4	16	1.00	393 (20.03)	3	9	1.00	532 (35.05)	4	16	1.00	461 (60.11)	4	16	1.00	1343 (30.02)	3	9	1.00	3007 (140.14)	5	16	1.25	958 (392.24)	8	49	1.57
Ω_5^{CC}	554 (46.26)	4	16	1.00	599 (140.39)	6	31	1.20	546 (123.56)	5	16	1.25	539 (480.28)	8	41	1.57	1686 (480.39)	8	33	1.57	3007 (140.15)	5	16	1.25	954 (480.11)	8	49	1.57
Ω_1^{OC}	554 (4.59)	4	16	1.00	498 (38.26)	4	16	1.00	546 (5.17)	5	16	1.25	566 (501.50)	7	43	2.00	1639 (460.75)	7	31	1.83	3007 (140.13)	5	16	1.25	888 (60.15)	4	16	1.00
Ω_2^{OC}	554 (2.22)	3	9	1.00	599 (86.35)	6	31	1.20	546 (7.76)	6	19	1.40	578 (495.54)	7	43	2.00	1280 (30.02)	3	9	1.00	3007 (140.24)	5	16	1.25	957 (511.19)	8	43	1.71
Ω_3^{OC}	554 (5.64)	3	9	1.00	599 (67.69)	6	31	1.20	546 (15.91)	5	16	1.25	444 (29.06)	3	9	1.00	1696 (601.38)	8	41	1.86	3007 (140.16)	5	16	1.25	824 (60.01)	4	16	1.00
Ω_4^{OC}	554 (3.57)	3	9	1.00	599 (57.99)	6	31	1.20	546 (9.18)	5	16	1.25	531 (121.65)	5	21	1.25	1209 (30.03)	3	9	1.00	3007 (140.15)	5	16	1.25	842 (60.02)	4	16	1.00
Ω_5^{OC}	554 (13.87)	4	16	1.00	599 (134.92)	6	31	1.20	546 (81.15)	5	16	1.25	447 (60.01)	4	16	1.00	1687 (580.45)	8	41	1.86	3007 (140.17)	5	16	1.25	954 (480.18)	8	49	1.57

Table F.1: Double incremental algorithm results. For each dataset and each configuration (Ω_1^K to Ω_5^K , for $K \in \{CC, OC\}$), we report their final objective value followed by the total time spent in minutes. Columns d^* and T^* report the final D and T found by the algorithm; column **avg** reports the average inside- T -loop iteration count.

monk1				monk2				monk3				balance-scale				car-evaluation				kr-kp				tic			
obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg	obj (time)	D^*	T^*	avg
Ω_1^{CC}	554 (19.35)	4	16	599 (75.31)	6	32	546 (8.86)	4	16	434 (30.08)	3	8	1686 (272.35)	8	106	3007 (100.12)	5	32	957 (270.34)	8	106						
Ω_2^{CC}	554 (16.43)	4	16	599 (77.37)	6	32	546 (46.77)	5	32	434 (30.09)	3	8	1343 (30.03)	3	8	3007 (100.12)	5	32	958 (270.65)	8	106						
Ω_3^{CC}	554 (24.97)	4	16	599 (79.68)	6	32	546 (41.88)	5	32	542 (270.37)	8	106	1691 (272.19)	8	106	3007 (100.16)	5	32	958 (260.38)	8	106						
Ω_4^{CC}	554 (21.42)	4	16	393 (22.14)	3	8	546 (59.68)	6	32	546 (269.71)	8	106	1343 (30.07)	3	8	3007 (100.13)	5	32	957 (270.35)	8	106						
Ω_5^{CC}	554 (43.32)	4	16	599 (100.40)	6	32	546 (49.46)	4	16	536 (270.20)	8	106	1686 (270.62)	8	106	3007 (100.15)	5	32	955 (270.21)	8	106						
Ω_1^{OC}	554 (1.90)	3	8	599 (69.44)	6	32	546 (4.71)	6	32	569 (262.81)	8	106	1686 (270.38)	8	106	3007 (100.16)	5	32	957 (210.22)	7	106						
Ω_2^{OC}	554 (1.90)	3	8	599 (58.43)	6	32	546 (14.30)	5	32	580 (248.20)	8	106	1391 (60.13)	4	16	3007 (100.19)	5	32	958 (178.89)	8	106						
Ω_3^{OC}	554 (4.86)	3	8	429 (15.21)	3	8	546 (8.69)	5	32	512 (121.27)	6	32	1686 (272.31)	8	106	3007 (100.14)	5	32	958 (170.86)	8	106						
Ω_4^{OC}	554 (4.68)	4	16	599 (64.42)	6	32	546 (5.58)	5	32	592 (252.23)	8	106	1209 (30.03)	3	8	3007 (100.28)	5	32	958 (220.24)	8	106						
Ω_5^{OC}	554 (2.16)	3	8	535 (96.64)	5	32	546 (75.91)	5	32	437 (60.07)	4	16	1686 (270.57)	8	106	3007 (100.20)	5	32	954 (270.18)	8	106						

Table F.2: Single incremental algorithm results with inequality constraints. For each dataset and each configuration (Ω_1^K to Ω_5^K , for $K \in \{CC, OC\}$), we report their final objective value followed by the total time spent in minutes. Columns D^* and T^* report the final D and T found by the algorithm.

	monk1			monk2			monk3			balance-scale			car-evaluation			kr-kp			tic		
	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*
Ω_1^{CC}	554 (46.68)	4	16	393 (21.11)	3	8	546 (12.23)	4	16	434 (60.14)	4	16	1686 (270.62)	8	256	3007 (100.17)	5	32	954 (270.13)	8	256
Ω_2^{CC}	554 (46.78)	4	16	393 (21.12)	3	8	546 (12.22)	4	16	434 (60.15)	4	16	1686 (270.77)	8	256	3007 (100.09)	5	32	954 (270.13)	8	256
Ω_3^{CC}	554 (36.77)	4	16	599 (90.08)	6	64	546 (68.76)	5	32	430 (30.04)	3	8	1686 (273.84)	8	256	3007 (100.08)	5	32	954 (270.08)	8	256
Ω_4^{CC}	554 (36.74)	4	16	599 (90.09)	6	64	546 (68.77)	5	32	430 (30.04)	3	8	1686 (272.62)	8	256	3007 (100.08)	5	32	954 (270.15)	8	256
Ω_5^{CC}	554 (49.04)	4	16	393 (20.78)	3	8	546 (69.87)	5	32	430 (30.02)	3	8	564 (30.03)	3	8	1527 (30.03)	3	8	670 (30.01)	3	8
Ω_1^{OC}	554 (4.58)	3	8	599 (65.04)	6	64	546 (5.23)	5	32	553 (253.29)	8	256	1686 (255.60)	8	256	3008 (100.16)	5	32	955 (270.29)	8	256
Ω_2^{OC}	554 (4.57)	3	8	599 (65.02)	6	64	546 (5.24)	5	32	553 (253.02)	8	256	1686 (270.82)	8	256	3008 (100.17)	5	32	955 (270.29)	8	256
Ω_3^{OC}	554 (7.65)	3	8	599 (65.50)	6	64	546 (24.03)	6	64	579 (258.66)	8	256	1635 (210.74)	7	128	3007 (100.08)	5	32	954 (270.30)	8	256
Ω_4^{OC}	554 (7.63)	3	8	599 (65.56)	6	64	546 (24.04)	6	64	579 (258.62)	8	256	1635 (210.72)	7	128	3007 (100.09)	5	32	954 (270.30)	8	256
Ω_5^{OC}	552 (5.31)	5	32	585 (241.69)	8	256	546 (41.18)	5	32	558 (266.76)	8	256	1280 (60.05)	4	16	2111 (30.03)	3	8	926 (270.54)	8	256

Table F.3: Single incremental algorithm results with equality constraints. For each dataset and each configuration (Ω_1^K to Ω_5^K , for $K \in \{CC, OC\}$), we report their final objective value followed by the total time spent in minutes. Columns D^* and T^* report the final D and T found by the algorithm.

	monk1			monk2			monk3			balance-scale			car-evaluation			kr-kp			tic		
	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*
DT	539 (25.98)	4	16	599 (94.62)	6	64	546 (69.93)	5	32	325 (330.01)	8	256	1439 (330.00)	4	16	2159 (330.02)	5	32	591 (330.04)	8	256
Ω_1^{CC}	554 (3.89)	4	16	599 (154.79)	6	64	546 (37.10)	5	32	239 (330.15)	8	256	1209 (330.22)	8	256	2111 (330.13)	5	32	652 (330.10)	8	256
Ω_2^{CC}	554 (8.05)	4	16	599 (319.70)	6	64	546 (234.67)	5	32	149 (330.12)	8	256	1243 (62.36)	4	16	2111 (330.20)	5	32	633 (330.11)	8	256
Ω_3^{CC}	554 (12.37)	4	16	599 (87.43)	6	64	546 (191.31)	6	64	288 (330.10)	8	256	1440 (79.10)	8	256	1815 (330.12)	5	32	626 (330.04)	8	256
Ω_4^{CC}	554 (6.79)	4	16	599 (169.54)	6	64	546 (330.08)	6	64	373 (27.68)	4	16	1028 (226.23)	3	8	2111 (330.13)	5	32	719 (330.13)	8	256
Ω_5^{CC}	554 (3.93)	4	16	393 (330.01)	6	64	546 (330.01)	5	32	450 (222.76)	3	8	284 (330.04)	4	16	1676 (330.10)	5	32	498 (330.14)	8	256
Ω_1^{OC}	554 (1.44)	4	16	599 (130.35)	6	64	546 (84.59)	6	64	365 (330.22)	8	256	1209 (330.22)	8	256	2111 (330.14)	5	32	946 (330.32)	7	128
Ω_2^{OC}	554 (1.71)	3	8	599 (79.68)	6	64	546 (46.97)	6	64	493 (331.12)	8	256	1243 (62.36)	4	16	2111 (330.09)	5	32	955 (330.07)	8	256
Ω_3^{OC}	554 (0.76)	3	8	599 (39.67)	6	64	546 (10.37)	5	32	603 (330.09)	6	64	1440 (79.10)	8	256	2113 (330.14)	5	32	950 (330.20)	8	256
Ω_4^{OC}	554 (1.91)	4	16	599 (38.07)	6	64	546 (6.18)	5	32	619 (331.24)	8	256	1419 (129.97)	3	8	2111 (330.12)	5	32	759 (330.13)	8	256
Ω_5^{OC}	554 (9.46)	4	16	393 (330.01)	6	64	546 (330.01)	5	32	515 (200.85)	4	16	0 (48.87)	8	256	2111 (330.10)	5	32	220 (330.11)	8	256

Table F.4: The results run without any incremental procedure without warm start with inequality constraints. The D^* is determined by taking the maximum of the single and double incremental procedure in each setting. T^* is set to be 2^{D^*} . All values are filled from the provided dataset.

	monk1			monk2			monk3			balance-scale			car-evaluation			kr-kp			tic		
	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*
DT	554 (4.83)	5	32	599 (147.81)	6	64	546 (79.73)	6	64	473 (178.41)	4	16	1172 (330.02)	8	256	2159 (330.02)	5	32	626 (330.04)	8	256
$\Omega^{CC}1$	-1 (330.10)	4	554	400 (330.15)	6	599	515 (330.05)	5	552	283 (330.12)	4	624	-1 (139.49)	4	16	-1 (330.13)	5	32	-1 (82.33)	8	958
$\Omega^{CC}2$	-1 (330.16)	4	554	400 (330.15)	6	599	-1 (330.15)	4	552	283 (330.12)	4	624	-1 (62.83)	3	8	-1 (330.20)	5	32	-1 (64.78)	8	958
$\Omega^{CC}3$	554 (144.39)	4	554	426 (330.15)	6	599	321 (330.16)	6	552	108 (321.36)	4	624	-1 (110.59)	4	16	-1 (330.12)	5	32	-1 (86.25)	8	958
$\Omega^{CC}4$	554 (144.65)	4	554	426 (330.15)	6	599	441 (343.92)	5	552	108 (323.34)	4	624	-1 (226.23)	3	8	-1 (330.13)	5	32	-1 (44.43)	8	958
$\Omega^{CC}5$	-1 (330.25)	4	554	-1 (330.18)	6	599	-1 (330.06)	5	552	-1 (196.59)	8	624	-1 (330.04)	4	16	-1 (330.10)	5	32	0 (27.46)	8	958
$\Omega^{OC}1$	554 (181.30)	4	554	330 (330.17)	6	599	526 (368.52)	5	552	432 (330.43)	8	624	-1 (330.22)	8	256	-1 (330.14)	5	32	0 (29.03)	7	958
$\Omega^{OC}2$	554 (162.29)	3	554	330 (330.17)	6	599	491 (330.09)	6	552	432 (330.48)	8	624	-1 (62.36)	4	16	-1 (330.09)	5	32	0 (29.36)	8	958
$\Omega^{OC}3$	537 (54.37)	3	554	597 (330.43)	6	599	546 (360.42)	6	552	549 (329.00)	8	624	-1 (79.10)	8	256	-1 (330.14)	5	32	0 (29.15)	8	958
$\Omega^{OC}4$	537 (54.50)	3	554	597 (330.77)	6	599	546 (367.89)	6	552	549 (325.99)	8	624	-1 (129.97)	3	8	-1 (330.12)	5	32	0 (29.04)	8	958
$\Omega^{OC}5$	554 (243.05)	5	554	17 (330.15)	8	599	544 (330.93)	5	552	10 (330.31)	8	624	-1 (48.87)	8	256	-1 (330.10)	5	32	0 (29.06)	8	958

Table F.5: Results without incremental procedures and without warm start, with equality constraints. For each configuration, the table reports the maximum objective value found (obj), the total solution time in seconds (time), the selected depth parameter D^* , and the corresponding threshold parameter T^* taken directly from the experimental results. A value of -1 indicates that no feasible solution was found within the time limit.

	monk1			monk2			monk3			balance-scale			car-evaluation			kr-kp			tic		
	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*
DT	539 (26.44)	4	16	599 (0.18)	6	64	546 (330.11)	5	32	462 (36.01)	3	8	1343 (330.57)	3	8	3007 (330.03)	5	32	954 (330.01)	8	256
Ω^{CC}	554 (2.15)	4	16	599 (0.61)	6	64	546 (37.87)	5	32	540 (330.18)	8	256	1209 (93.32)	4	16	3007 (15.16)	5	32	955 (330.05)	8	256
Ω^{OC}	554 (5.67)	4	16	599 (0.57)	6	64	546 (34.40)	5	32	545 (330.06)	8	256	1343 (100.57)	3	8	3007 (14.64)	5	32	957 (330.08)	8	256
Ω^{OC}	554 (5.20)	4	16	599 (0.51)	6	64	546 (30.62)	6	64	552 (330.22)	8	256	1209 (120.74)	4	16	3007 (169.28)	5	32	958 (330.22)	8	256
Ω^{OC}	554 (21.80)	4	16	599 (0.49)	6	64	546 (330.13)	6	64	408 (12.21)	4	16	1065 (184.17)	3	8	3007 (14.95)	5	32	958 (286.75)	8	256
Ω^{OC}	554 (0.66)	4	16	599 (0.69)	6	64	546 (330.13)	5	32	467 (330.01)	3	8	284 (330.04)	4	16	3007 (26.14)	5	32	954 (330.16)	8	256
Ω^{OC}	554 (2.44)	4	16	599 (0.80)	6	64	546 (89.77)	6	64	536 (330.18)	8	256	-1 (0.70)	7	128	3007 (15.14)	5	32	955 (330.05)	8	256
Ω^{OC}	554 (10.53)	4	16	599 (0.80)	6	64	546 (49.13)	6	64	570 (330.21)	8	256	1279 (330.02)	3	8	3007 (14.98)	5	32	957 (330.08)	8	256
Ω^{OC}	554 (5.67)	4	16	599 (0.77)	6	64	546 (1.04)	5	32	600 (330.11)	6	64	1492 (260.24)	3	8	3007 (14.94)	5	32	958 (55.24)	8	256
Ω^{OC}	554 (5.20)	4	16	599 (0.77)	6	64	546 (0.84)	5	32	613 (330.26)	8	256	1501 (330.20)	3	8	3007 (15.03)	5	32	958 (56.66)	8	256
Ω^{OC}	554 (21.80)	4	16	599 (0.97)	6	64	546 (244.85)	5	32	535 (237.01)	4	16	-	-	-	3007 (22.74)	5	32	954 (330.13)	8	256

Table F.6: The results run without any incremental procedure with warm start from CART, using inequality constraints. The D^* is determined by taking the maximum of the single and double incremental procedure in each setting. T^* is set to be 2^{D^*} .

	monk1			monk2			monk3			balance-scale			car-evaluation			kr-kp			tic		
	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*	obj (time)	D^*	T^*
DT	554 (3.48)	5	32	599 (0.09)	6	64	546 (330.03)	6	64	464 (159.95)	4	16	1686 (330.02)	8	256	3007 (330.02)	5	32	954 (330.01)	8	256
Ω_1^{CC}	-1 (330.10)	4	554	599 (35.24)	6	599	546 (330.20)	5	552	283 (330.12)	4	624	-1 (139.49)	4	16	-1 (330.13)	5	32	954 (335.90)	8	958
Ω_2^{CC}	471 (330.26)	4	554	599 (35.20)	6	599	506 (330.34)	4	552	283 (330.12)	4	624	-1 (62.83)	3	8	-1 (330.20)	5	32	954 (38.71)	8	958
Ω_3^{CC}	554 (132.55)	4	554	599 (35.65)	6	599	546 (330.05)	6	552	108 (295.06)	4	624	-1 (110.59)	4	16	-1 (330.12)	5	32	954 (75.45)	8	958
Ω_4^{CC}	554 (133.50)	4	554	599 (35.40)	6	599	546 (339.16)	5	552	108 (287.71)	4	624	-1 (226.23)	3	8	-1 (330.12)	5	32	954 (74.98)	8	958
Ω_5^{CC}	-1 (330.27)	4	554	-1 (330.17)	6	599	517 (330.06)	5	552	-1 (192.90)	8	624	-1 (330.04)	4	16	-1 (330.10)	5	32	-1 (365.28)	8	958
Ω_1^{OC}	554 (125.81)	4	554	599 (35.81)	6	599	546 (370.96)	5	552	563 (330.34)	8	624	-1 (330.22)	8	256	-1 (330.14)	5	32	958 (74.32)	7	958
Ω_2^{OC}	554 (236.91)	3	554	599 (35.81)	6	599	546 (330.05)	6	552	563 (330.35)	8	624	-1 (62.36)	4	16	-1 (330.09)	5	32	958 (65.41)	8	958
Ω_3^{OC}	554 (63.79)	3	554	599 (36.02)	6	599	546 (392.54)	6	552	583 (328.94)	8	624	-1 (79.10)	8	256	-1 (330.14)	5	32	958 (65.38)	8	958
Ω_4^{OC}	554 (63.92)	3	554	599 (35.95)	6	599	546 (392.82)	6	552	583 (330.16)	8	624	-1 (129.97)	3	8	-1 (330.12)	5	32	958 (65.34)	8	958
Ω_5^{OC}	554 (233.34)	5	554	17 (330.24)	8	599	541 (330.21)	5	552	10 (330.30)	8	624	-1 (48.87)	8	256	-1 (330.10)	5	32	958 (65.25)	8	958

Table F.7: The results run without any incremental procedure with warm start from **CART**, with equality constraints. The D^* is determined by taking the maximum of the single and double incremental procedure in each setting. T^* is set to be $\min(I, \binom{N}{D} 2^{D^*})$.

G QUBO Numerical Results

In this section, we present the detailed numerical results obtained for the D-Wave hybrid quantum annealer¹⁴ solving the QUBO formulations from Section 3.3, under the *single incremental* (SI-QA) scheme. Our reliance on the hybrid solver, rather than on direct access to the physical annealer, is a consequence of the scale of the instances considered. Running the problems on the bare quantum processor would require expressing each instance as a Binary Quadratic Model (BQM) submitted directly to the QPU, which is the only mode in which the user retains full control over the annealing schedule and, hence, over the adiabatic evolution. This route is infeasible here: as of 2026, the largest quantum annealer in existence provides at most on the order of 5,000 qubits, far too few to embed the QUBOs associated with any of the datasets considered in this work, especially once the overhead of minor-embedding is taken into account. Because present-day hybrid annealers impose a hard cap of 100,000 constraints, the larger datasets cannot be encoded in full at every depth; we therefore report results under two complementary protocols, summarized in the following artifacts:

- Table G.1 reports the results obtained on the full datasets, for the configurations that fit under the D-Wave hybrid solver’s 100,000-constraint cap.
- Table G.2 reports the results obtained on a subsampled protocol, where 6 random 300-row subsamples (seeds 40, 42, 44, 46, 48, 50) are drawn for each dataset and both SI-QA and CART are re-run on the same subsamples.
- Figure 8 displays, for each dataset, the per-depth distribution of SI-QA and CART objective values obtained under the subsampled protocol.

The tables and figure are organized around the two Ω equivalence classes $\{\Omega_1 \equiv \Omega_2 \equiv \Omega_5\}$ and $\{\Omega_3 \equiv \Omega_4\}$ established in Section 3.1, and the two classification modes $K \in \{\text{CC}, \text{OC}\}$. For each dataset–configuration pair we report the objective value Z (number of correctly classified instances), the depth D^* at which it is obtained, the average running time, and the QUBO problem size in terms of binary variables n_{var} and constraints n_{constr} . In Table G.1, D^* is the largest depth that fits under the constraint cap; in Table G.2 and Figure 8, D^* is the depth that maximizes the mean objective across the 6 subsamples, and CART is evaluated on the same subsamples as the paired heuristic baseline. Cells marked “—” correspond to configurations for which no depth fits.

¹⁴The D-Wave hybrid solver, also known as the Constrained Quadratic Model (CQM) solver, manages internally all parameters governing the adiabatic evolution of the annealer. The precise manner in which these parameters are set is proprietary, and the user has no possibility of tuning them.

dataset	$\Omega_{1\equiv 2\equiv 5}^{\text{CC}}$				$\Omega_{3\equiv 4}^{\text{CC}}$			
	D^*	Z	n_{var}	n_{constr}	D^*	Z	n_{var}	n_{constr}
monk1	3	350	7,422	74,246	3	347	7,422	74,356
monk2	2	393	6,765	64,700	2	393	6,765	64,788
monk3	2	349	6,248	59,624	2	358	6,248	59,712
balance-scale	1	348	6,432	70,518	1	366	6,432	70,614
car-evaluation	-	-	-	>100,000	-	-	-	>100,000
kr-kp	-	-	-	>100,000	-	-	-	>100,000
tic	1	637	6,850	80,476	1	627	6,850	80,548

(a) Categorical classification.

dataset	$\Omega_{1\equiv 2\equiv 5}^{\text{OC}}$				$\Omega_{3\equiv 4}^{\text{OC}}$			
	D^*	Z	n_{var}	n_{constr}	D^*	Z	n_{var}	n_{constr}
monk1	3	382	7,422	73,968	3	414	7,422	74,078
monk2	2	393	6,765	64,307	2	393	6,765	64,395
monk3	2	387	6,248	59,358	2	440	6,248	59,446
balance-scale	1	323	6,432	70,518	1	383	6,432	70,614
car-evaluation	-	-	-	>100,000	-	-	-	>100,000
kr-kp	-	-	-	>100,000	-	-	-	>100,000
tic	1	613	6,850	80,144	1	583	6,850	80,216

(b) Ordinal classification.

Table G.1: Full-dataset SI-QA results. For each dataset, mode, and Ω equivalence class, we report the maximum depth D^* for which the resulting QUBO fits under the D-Wave hybrid solver 10^5 -constraint cap, together with the objective value Z (number of correctly classified instances), the number of binary variables n_{var} , and the number of constraints n_{constr} at that depth. Cells marked “–” correspond to configurations for which no depth fits.

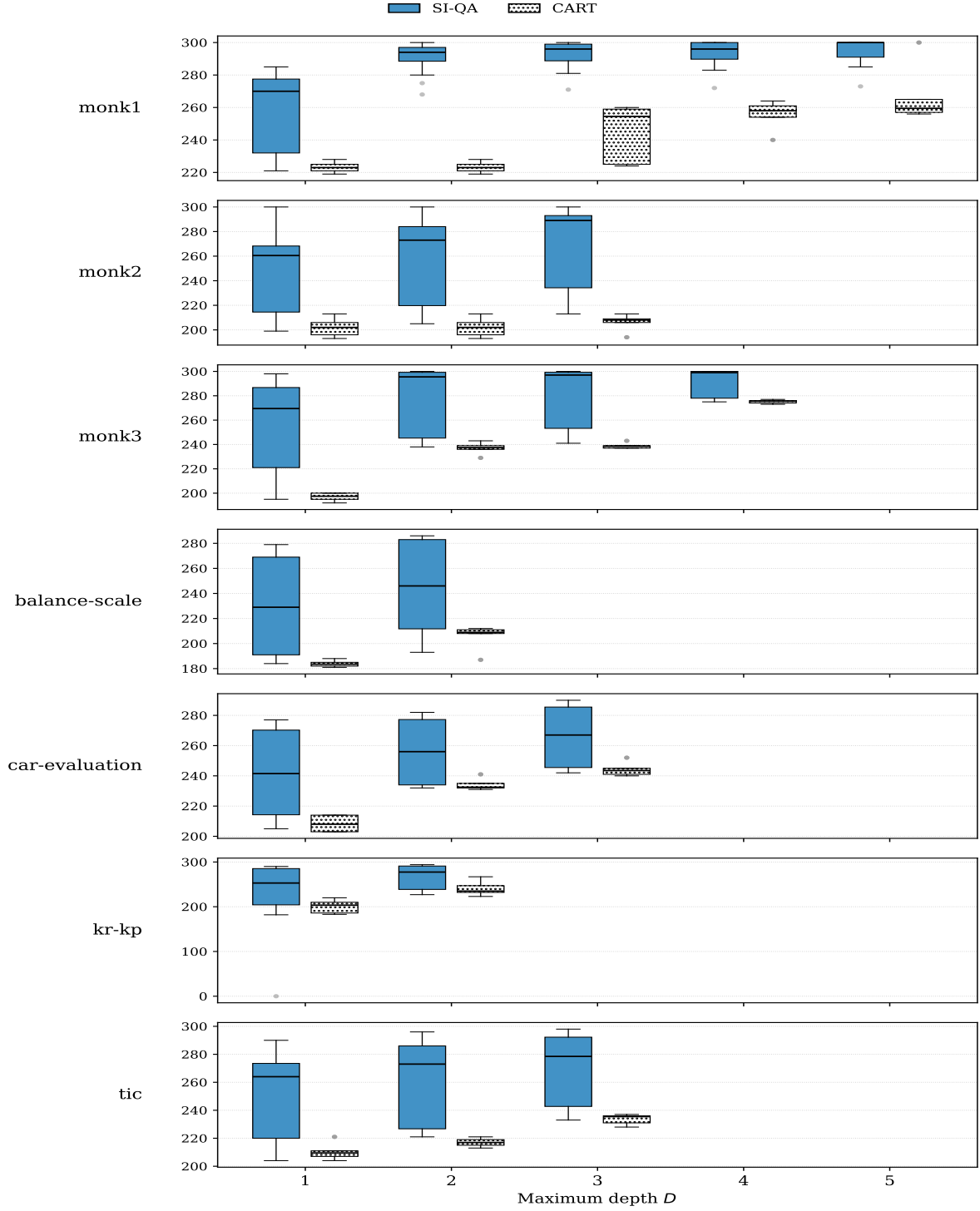


Figure 8: Per-dataset distribution of correctly classified instances as a function of the maximum DNF clause depth D under the subsampled protocol, comparing SI-QA (solid blue) with CART (hatched). Each row corresponds to one dataset, and the vertical axis reports the number of correctly classified instances on a 300-row subsample. At each depth, the SI-QA box aggregates the 24 observations obtained from the 6 random subsamples (seeds 40, 42, 44, 46, 48, 50), the two modes $K \in \{CC, OC\}$, and the two Ω equivalence classes $\{\Omega_1 \equiv \Omega_2 \equiv \Omega_5, \Omega_3 \equiv \Omega_4\}$; the CART box aggregates the corresponding 12 observations on the same subsamples and modes, since CART does not depend on the Ω class. Depth values for which no SI-QA run returned a feasible solution under the D-Wave hybrid solver’s 100,000 constraint cap are omitted from the panel.

H Oversampling

As discussed in Section 3.2, a **CART** warm start may be sample-degenerate, meaning that at least one class is never predicted on the training sample. This situation is inconvenient when the tree is converted into DNF form, because the DNF associated with a sample-empty class must be represented by clauses that are false on the training observations, unless the formulation explicitly neutralizes empty clauses.

Under the mild condition that the number of leaves is at least the number of classes, i.e., $2^D \geq |\mathcal{C}|$, one can always construct a class-complete leaf assignment by relabeling some leaves so that every class is assigned to at least one leaf. This naive repair, however, may degrade the training objective more than necessary, especially when a relabeled leaf contains many observations from another class. We therefore use the oversampling heuristic described in Algorithm 1 below. The purpose of the procedure is to encourage **CART** to produce a non-degenerate classifier by duplicating observations from classes that are currently not predicted.

Remark 3 (Oversampling procedure). *The oversampling procedure attempts to construct a sample-non-degenerate **CART** solution within a prescribed maximum number of iterations. If no such solution is obtained, the procedure defaults to the naive leaf-relabeling repair. The resulting tree can then be converted into a DNF warm start using Definition 4, and, when needed, saturated using Theorem 4.*

Algorithm 1 Oversampling procedure for non-degenerate solutions

- 1: Data: original training sample $(\mathbf{v}_i, u_i)$ for $i = 1, \dots, I$, maximum iteration count `max_iter`.
- 2: Initialize $\mathcal{I}_0 \leftarrow \{1, \dots, I\}$ and $n \leftarrow 0$.
- 3: Solve **CART**(D) on the current sample $(\mathbf{v}_i, u_i)_{i=1}^{I+n}$, and let $U^{(n)}$ be the resulting classifier.
- 4: Compute the set of sample-empty classes on the original sample:

$$\mathcal{C}^0(U^{(n)}) = \left\{ c \in \mathcal{C} : U^{(n)}(\mathbf{v}_i) \neq c, \forall i \in \mathcal{I}_0 \right\}.$$

- 5: **while** $\mathcal{C}^0(U^{(n)}) \neq \emptyset$ and $n < \text{max_iter}$ **do**
- 6: Select a class $c^* \in \mathcal{C}^0(U^{(n)})$.
- 7: Select an observation $i^* \in \mathcal{I}_0$ such that $u_{i^*} = c^*$.
- 8: Set $n \leftarrow n + 1$.
- 9: Append a duplicate observation $(\mathbf{v}_{I+n}, u_{I+n}) := (\mathbf{v}_{i^*}, u_{i^*})$ to the current sample.
- 10: Solve **CART**(D) on the augmented sample $(\mathbf{v}_i, u_i)_{i=1}^{I+n}$, and let $U^{(n)}$ be the resulting classifier.
- 11: Update

$$\mathcal{C}^0(U^{(n)}) = \left\{ c \in \mathcal{C} : U^{(n)}(\mathbf{v}_i) \neq c, \forall i \in \mathcal{I}_0 \right\}.$$

- 12: **end while**
 - 13: **if** $\mathcal{C}^0(U^{(n)}) = \emptyset$ **then**
 - 14: **return** the **CART** solution $U^{(n)}$.
 - 15: **else**
 - 16: **return** the naive leaf-relabeling solution.
 - 17: **end if**
-

I Detailed incremental procedures

This appendix reports the detailed pseudo-code of the two incremental procedures described in Section 4. Algorithm 2 corresponds to the single incremental procedure (SI), where only the complexity parameter D is increased. Algorithm 3 corresponds to the double incremental procedure (DI), where both D and the DNF clause budget T are increased.

The notation **BestVal** denotes the best incumbent objective value found so far, while **BestSolution** denotes the corresponding feasible classifier. The notation **CARTVal**(D) denotes the objective value of the **CART** solution obtained at a given depth D . In the DNF case, **COC**(D, T) denotes the categorical or ordinal classification problem based on formulation (4) or (11), with literal depth D and clause budget T .

The single incremental procedure is used when the structural size of the model is determined once D is fixed. This is the case for DT formulations, where a full tree of depth D has 2^D leaves. It is also the case for exact- D saturated DNF or QUBO formulations when the number of clauses is fixed according to a predetermined clause schedule, such as $T(D) = 2^D$ or $T_{\text{sat}}(D) = T_{[=D]} = \min\{I, \binom{F}{D}2^D\}$.

Algorithm 2 Single incremental procedure for DT or exact- D saturated DNF/QUBO formulations

```

1: Initialize BestVal  $\leftarrow 0$  and BestSolution  $\leftarrow \emptyset$ .
2: for  $D = 2, \dots, D_{\max}$  do
3:   Solve CART( $D$ ) and store its objective value in CARTVal( $D$ ).
4:   if CARTVal( $D$ ) =  $I$  then
5:     return CART( $D$ ).
6:   end if
7:   Warm start the selected formulation with the incumbent obtained at the previous value of
      $D$  or with CART( $D$ ); in exact- $D$  DNF/QUBO formulations, lift the previous incumbent using
     Lemma 1.
8:   Solve the selected formulation at complexity level  $D$  and store its objective value in SIVal( $D$ ).
9:   if SIVal( $D$ ) > BestVal then
10:    BestVal  $\leftarrow$  SIVal( $D$ ) and update BestSolution.
11:  end if
12:  if BestVal =  $I$  then
13:    return BestSolution.
14:  end if
15: end for
16: return BestSolution.

```

The logic of Algorithm 2 is deliberately conservative. At each iteration, the procedure compares the **CART** incumbent with the best solution already obtained. If the **CART** solution is better, it is used as the warm start for the current solve. Otherwise, the previous incumbent is used. In exact- D DNF or QUBO formulations, this incumbent must first be lifted to the new value of D using Lemma 1. The procedure stops immediately if a feasible solution with objective value equal to I (number of

observations in the training dataset) is found, since this means a perfect classification.

As a numerical tuning to balance solution quality and computational effort, in our numerical tests in Section 5, the MIP optimality gap at each iteration of Algorithm 2 is tightened as D increases, while the per-solve time limit is increased, as reported in Table I.1.

D	2	3	4	5	6	7	8	9
MIPGap	0.1	0.05	0.01	0.005	0.001	0.0005	0.0001	0.0001
Time limit	600	1200	1800	2400	3000	3600	3600	3600

Table I.1: Settings for D , MIPGap, and per-solve time limits (in seconds).

The double incremental procedure is specific to DNF-based formulations. For each fixed value of D , it solves a sequence of formulations with increasing clause budgets T . This strategy exploits the fact that DNF formulations allow the number of available clause slots to be controlled independently of the maximum number of literals per clause. In contrast, DT formulations do not admit this additional degree of freedom, since their number of leaves is determined by the depth of the tree.

Algorithm 3 Double incremental procedure for DNF/QUBO formulations (4) or (11)

```

1: Initialize BestVal  $\leftarrow 0$  and BestSolution  $\leftarrow \emptyset$ .
2: for  $D = 2, \dots, D_{\max}$  do
3:   Solve CART( $D$ ) and store its objective value in CARTVal( $D$ ).
4:   if CARTVal( $D$ ) =  $I$  then
5:     return CART( $D$ ).
6:   end if
7:   Compute  $T_{\text{WS}}(D)$  from the DT-to-DNF mapping.
8:   Set  $T_{\min}(D) = \max\{2D, T_{\text{WS}}(D)\}$ .
9:   Choose  $T_{\max}(D)$  according to the selected truncated dictionary.
10:  for  $T = T_{\min}(D), \dots, T_{\max}(D)$  do
11:    Construct a warm start from the best available feasible solution among:
12:      the mapped solution of CART( $D$ );
13:      the solution of COC( $D, T - 1$ ), if  $T > T_{\min}(D)$ ;
14:      the best solution obtained at smaller values of  $D$ , embedded in the current  $(D, T)$  formulation when feasible.
15:    Solve COC( $D, T$ ) and store its objective value in COCVal( $D, T$ ).
16:    if COCVal( $D, T$ ) > BestVal then
17:      BestVal  $\leftarrow$  COCVal( $D, T$ ) and update BestSolution.
18:    end if
19:    if BestVal =  $I$  then
20:      return BestSolution.
21:    end if
22:  end for
23: end for
24: return BestSolution.

```

In Algorithm 3, the value $T_{\text{WS}}(D)$ is obtained by converting the **CART** solution into DNF form. For the non-saturated use of Algorithm 3, these path clauses are counted before any exact- D saturation

step; if an exact- D saturated variant is used instead, the count is computed after saturation and T is fixed by D as in Algorithm 2. The lower bound $T_{\min}(D) = \max\{2D, T_{\text{WS}}(D)\}$ ensures that the inner loop starts from a clause budget large enough to encode the available tree-based warm start. The upper bound $T_{\max}(D)$ is selected according to the intended truncation of the DNF dictionary. In the non-saturated case, one requires $T_{\max}(D) \leq \sum_{k=0}^D \binom{F}{k} 2^k$. In the saturated exact- D case, one requires $T_{\max}(D) \leq T_{\text{sat}}(D)$.

As in the single incremental procedure, Algorithm 3 keeps the best feasible solution found so far. This is important because, although the exact optimal value is nondecreasing in both D and T under the relevant embedding arguments, the incumbent values returned by a solver under time limits or positive optimality gaps need not be monotone.