Decision Sets Paper: Comprehensive Exam with Solutions

PROBLEM-SOLVING QUESTIONS

Question 1: Rule Enumeration Design

Difficulty: Medium

Consider a binary classification problem with 3 features (f_1, f_2, f_3) and 5 positive examples in $E^{\oplus}$:

- $e_1 = (1, 0, 1) \oplus$
- $e_2 = (1, 0, 1) \oplus$
- $e_3 = (0, 1, 1) \oplus$
- $e_4 = (1, 1, 0) \oplus$
- $e_5 = (0, 0, 1) \oplus$

And 3 negative examples in $E^{\ominus}$:

- $e_6 = (1, 1, 1) \ominus$
- $e_7 = (0, 1, 0) \ominus$
- $e_8 = (1, 0, 0) \ominus$

Task: Using the dual-rail encoding described in Section 4.2, construct the hard clauses (equations 2, 4, and 6) that would be needed to enumerate a single valid rule for class $\oplus$. You do not need to write the soft clauses.

Solution:

Using dual-rail variables: p_1, n_1, p_2, n_2, p_3, n_3 (where $p_r = 1$ iff $f_r = 1$, $n_r = 1$ iff $f_r = 0$)

Hard Clause Set (Equation 2) - Forbid dual values:

$$(\neg p_1 \vee \neg n_1) \wedge (\neg p_2 \vee \neg n_2) \wedge (\neg p_3 \vee \neg n_3)$$

Hard Clause Set (Equation 4) - Discriminate all $E^{\ominus}$ examples:

For $e_6 = (1, 1, 1)$: To discriminate, need at least one literal that doesn't match: $(n_1 \vee n_2 \vee n_3)$
For $e_7 = (0, 1, 0)$: To discriminate, need: $(p_1 \vee n_2 \vee p_3)$
For $e_8 = (1, 0, 0)$: To discriminate, need: $(n_1 \vee p_2 \vee p_3)$

Combined: $(n_1 \vee n_2 \vee n_3) \wedge (p_1 \vee n_2 \vee p_3) \wedge (n_1 \vee p_2 \vee p_3)$

Hard Clause Set (Equation 6) - Cover at least one $E\oplus$ example:

First, define auxiliary variables:

- $t_1 \leftrightarrow \neg(n_1 \vee p_2 \vee n_3)$ [covers $e_1 = (1,0,1)$ if no discriminating literals]
- $t_2 \leftrightarrow \neg(n_1 \vee p_2 \vee n_3)$ [covers $e_2 = (1,0,1)$, same as e_1]
- $t_3 \leftrightarrow \neg(p_1 \vee n_2 \vee n_3)$ [covers $e_3 = (0,1,1)$]
- $t_4 \leftrightarrow \neg(n_1 \vee n_2 \vee p_3)$ [covers $e_4 = (1,1,0)$]
- $t_5 \leftrightarrow \neg(p_1 \vee p_2 \vee n_3)$ [covers $e_5 = (0,0,1)$]

Final clause: $(t_1 \vee t_2 \vee t_3 \vee t_4 \vee t_5)$

Question 2: Set Cover Formulation

Difficulty: Medium

After enumerating rules for a dataset, you obtain $T\oplus = \{\pi_1, \pi_2, \pi_3, \pi_4\}$ where:

- $\pi_1 = (f_1 \wedge f_2)$: covers examples $\{e_1, e_2, e_5\}$
- $\pi_2 = (f_1)$: covers examples $\{e_1, e_2, e_3, e_5\}$
- $\pi_3 = (\neg f_2)$: covers examples $\{e_2, e_4, e_5\}$
- $\pi_4 = (f_1 \wedge \neg f_2 \wedge f_3)$: covers examples $\{e_1, e_2\}$

Assume $E\oplus = \{e_1, e_2, e_3, e_4, e_5\}$ (5 positive examples).

Task: Formulate the ILP set cover problem (equations 7-8) to minimize the number of rules. Which solution is optimal?

Solution:

ILP Formulation:

Decision variables: $b_1, b_2, b_3, b_4 \in \{0, 1\}$

Coverage matrix ($a_{ij} = 1$ if π_j covers e_i):

	π_1	π_2	π_3	π_4
e_1	1	1	0	1
e_2	1	1	1	1
e_3	0	1	0	0
e_4	0	0	1	0
e_5	1	1	1	0

Objective (Equation 7):

$$\text{Minimize: } b_1 + b_2 + b_3 + b_4$$

Constraints (Equation 8):

$$b_1 + b_2 + 0 \cdot b_3 + b_4 \geq 1 \quad (e_1)$$
$$b_1 + b_2 + b_3 + b_4 \geq 1 \quad (e_2)$$
$$0 \cdot b_1 + b_2 + 0 \cdot b_3 + 0 \cdot b_4 \geq 1 \quad (e_3)$$
$$0 \cdot b_1 + 0 \cdot b_2 + b_3 + 0 \cdot b_4 \geq 1 \quad (e_4)$$
$$b_1 + b_2 + b_3 + 0 \cdot b_4 \geq 1 \quad (e_5)$$

Simplified:

$$b_1 + b_2 + b_4 \geq 1$$
$$b_1 + b_2 + b_3 + b_4 \geq 1$$
$$b_2 \geq 1$$
$$b_3 \geq 1$$
$$b_1 + b_2 + b_3 \geq 1$$

Solution Analysis: From constraint 3: $b_2 = 1$ (required) From constraint 4: $b_3 = 1$ (required)

With $b_2 = 1$ and $b_3 = 1$:

- Constraint 1: satisfied ($b_2 = 1$)
- Constraint 2: satisfied ($b_2 + b_3 = 2$)
- Constraint 5: satisfied ($b_2 + b_3 = 2$)

Optimal solution: $b_2 = 1$, $b_3 = 1$, $b_1 = 0$, $b_4 = 0$ **Minimum number of rules:** 2 (select π_2 and π_3) **Selected decision set:** $\pi_2 = (f_i)$ and $\pi_3 = (\neg f_2)$

Question 3: Handling Overlapping Rules

Difficulty: Hard

In Example 7 of the paper, they mention symmetric rules $\pi_1 = (f_1 \wedge f_3)$ and $\pi_2 = (f_1 \wedge \neg f_4)$ both cover the same positive examples in $E\oplus$.

Task: a) Explain why having symmetric rules is problematic for the set cover problem. b) Describe the symmetry-breaking method from Section 4.3 and write the hard clause that would be added to prevent finding π_2 if π_1 was already discovered. c) Would this symmetry-breaking method affect optimality? Justify.

Solution:

a) Why symmetric rules are problematic:

Symmetric rules cover identical sets of examples. In the set cover problem, only one rule from a set of symmetric rules can be selected in an optimal solution (since minimizing requires picking the fewest rules). However, without breaking symmetry, the MaxSAT enumeration phase will continue finding all symmetric variants, causing an exponential explosion of rules that must be stored and processed. This wastes computational resources on redundant rules.

From the paper's experiments: without symmetry breaking, average enumerated rules = 19,604.4; with symmetry breaking = 563.7 (35x reduction).

b) Symmetry-breaking method:

After discovering rule π_1 with coverage $E'\oplus \subset E\oplus$, add the hard clause to H:

$$(\bigvee_i \in E\oplus \backslash E'\oplus\ t_i)$$

This enforces that any new rule discovered must cover at least one example NOT covered by π_1.

Concrete example: If π_1 covers $\{e_2, e_3\}$, and $E\oplus = \{e_1, e_2, e_3, e_4\}$, the clause added is:

$$(t_1 \vee t_4)$$

This forces the next rule to cover either e_1 or e_4 (or both), preventing discovery of another symmetric rule covering only $\{e_2, e_3\}$.

c) Effect on optimality:

No, optimality is NOT affected. The symmetry-breaking only prevents enumeration of symmetric variants—it does not prevent discovering an optimal solution because:

1. By definition, symmetric rules produce identical set cover solutions (they cover the same examples)

2. If multiple symmetric rules exist, selecting any one of them in a solution is equally optimal
3. The method only eliminates the redundant symmetric copies; at least one representative remains

Therefore, the minimum-size set cover is still guaranteed to exist and be found with the symmetry-breaking constraints applied.

ANALYSIS AND PROOF QUESTIONS

Question 4: Correctness of Rule Enumeration

Difficulty: Hard

The paper claims (after equation 6) that "as soon as exhaustive solution enumeration for formula (1) is finished, the set of terms $T\oplus$ covers every example $e_i \in E\oplus$."

Task: Prove this claim. Specifically: a) Show that for every example $e_i \in E\oplus$, there exists at least one term $\pi \in T\oplus$ that covers e_i. b) What is the key assumption required for this claim to hold?

Solution:

a) Proof:

Consider any example $e_i \in E\oplus$ with feature values v_i.

By assumption, E and $E\oplus$ are perfectly classifiable (non-overlapping), so no example in E has the same feature values as e_i.

Claim: There exists a term π such that:

1. π covers e_i (i.e., no literal in π discriminates e_i)
2. π discriminates all examples in E
3. π is irreducible (minimal)

Construction: Consider the following partial MaxSAT problem:

- Set all dual-rail variables such that $p_r = v_{ir}$ and $n_r = 1 - v_{ir}$ for all features
- This assignment makes auxiliary variable $t_i = 1$ (covering e_i)
- By equation 4, all constraints discriminating E are either already satisfied (tautologies for this assignment when e_i's features differ from E examples) or force certain literals into the term

Since training data is non-overlapping, for each example $e_j \in E$, there exists at least one feature where e_j differs from e_i. This guarantees equation 4 can be satisfied.

By exhaustive enumeration in the MaxSAT phase, we will find at least one term that:

- Satisfies equation 6 (covers e_i via $t_i = 1$)
- Satisfies all hard clauses (equations 2, 4)
- Minimizes soft clauses (is minimal)

Therefore, for every $e_i \in E\oplus$, at least one term in $T\oplus$ covers it.

b) Key assumption:

The critical assumption is that **the training data E is perfectly classifiable (non-overlapping)**. This means:

- No two examples have identical feature values but different class labels
- Equivalently, $E\oplus \cap E = \emptyset$ when considering feature values

Without this assumption, there could exist an example $e_i \in E\oplus$ whose feature values are identical to some $e_j \in E$. In that case, any term covering e_i would automatically also satisfy (not discriminate) e_j, making it impossible to satisfy the discrimination constraints (equation 4) for that e_j.

The paper addresses this: "we assume wlog. that the training data E is perfectly classifiable...in other words, there are no overlapping examples in E."

Question 5: Complexity Analysis

Difficulty: Hard

Compare the encoding complexity of the proposed approach versus the SAT-based methods of Ignatiev et al. (2018) and Yu et al. (2020).

Task: a) What is the asymptotic complexity (in terms of variables and clauses) for the old approach? For the new approach? b) Why does the two-stage approach achieve better complexity? c) When might the old approach be preferable despite worse asymptotic complexity?

Solution:

a) Asymptotic complexity comparison:

Old SAT-based approach (Ignatiev et al. 2018, Yu et al. 2020):

- Variables: $O(N \times M \times K)$

- Clauses: O(N × M × K)

Where:

- N = target size of decision set (number of rules or literals to determine)
- M = number of training examples
- K = number of features

New two-stage approach:

Phase 1 (Rule enumeration):

- Variables: O(K + M)
- Clauses: O(K × M)

Phase 2 (Set cover):

- Variables: O(L) where L = |T⊕| + |T⊖| ≤ 2^K (worst case exponential, but usually much smaller in practice)
- Clauses: O(L × M)

Total: O(K × M) + O(L × M) dominated by O(L × M), but crucially L is not predetermined and often much smaller than N × M × K.

b) Why two-stage is better:

1. **Decoupling:** Phase 1 doesn't need to know the target decision set size N in advance. It generates candidate rules incrementally.
2. **Smaller initial encoding:** Phase 1 has O(K + M) variables instead of O(N × M × K), making each MaxSAT call simpler.
3. **Amenability to ILP:** Phase 2 is a pure set cover problem, which ILP solvers handle very effectively (better than general SAT encodings).
4. **Practical L is small:** While L can theoretically be exponential in K, in practice L is much smaller than N × M × K because:
   - Irreducibility constraints (Section 4.2) prevent redundant rules
   - Symmetry breaking (Section 4.3) eliminates redundant symmetric variants
   - Real-world datasets don't generate exponentially many distinct minimal rules

From experiments: average L = 563.7 (with symmetry breaking) versus theoretical worst case 2^K.

c) When might old approach be preferable:

1. **Very small datasets:** When M and K are tiny, the simpler all-in-one encoding might solve faster without enumeration overhead.
2. **Sparse solutions:** When the optimal decision set is already very small (N is tiny), the old approach's O(N × M × K) might be smaller than the enumeration phase's overhead.
3. **Highly specific rules:** In domains where meaningful rules are very specific and numerous, L might approach N × M × K, negating the complexity advantage.
4. **Hardware constraints:** If memory for storing all enumerated rules $T_{\oplus}$ becomes a bottleneck.

However, the experimental results show the proposed approach wins on 802/1065 benchmarks versus 351 for the best old approach, suggesting the two-stage method is superior in practice for realistic datasets.

Question 6: DNF Representation Correctness

Difficulty: Medium

The paper represents decision sets as DNF (disjunctive normal form) formulas $\varphi_{\oplus}$ and $\varphi_{\ominus}$.

Task: a) In Example 2, verify that the DNF formulas $\varphi_{\ominus} = (f_4) \vee (\neg f_1)$ and $\varphi_{\oplus} = (\neg f_4 \wedge f_1)$ correctly represent the decision set from Example 1. Check against all four training examples.

b) Can a decision set always be losslessly converted to a DNF formula? Why or why not?

Solution:

a) Verification:

From Example 1, the decision set is:

- Rule 1: IF TV Show = Good THEN Date = No
- Rule 2: IF Day = Weekday THEN Date = No
- Rule 3: IF TV Show = Bad ∧ Day = Weekend THEN Date = Yes

Mapping (from Example 2):

- f_1 = Day (0=Weekday, 1=Weekend)
- f_2 = Venue (0=Dinner, 1=Club)
- f_3 = Weather (0=Cold, 1=Warm)
- f_4 = TV Show (0=Good, 1=Bad)

- $\ominus$ = No, $\oplus$ = Yes

Translating the decision set:

- Rule 1: $(f_4 = 0) \rightarrow \ominus$ becomes $(\neg f_4) \rightarrow \ominus$
- Rule 2: $(f_1 = 0) \rightarrow \ominus$ becomes $(\neg f_1) \rightarrow \ominus$
- Rule 3: $(f_4 = 1 \wedge f_1 = 1) \rightarrow \oplus$ becomes $(f_4 \wedge f_1) \rightarrow \oplus$

Therefore:

- $\varphi\ominus = (\neg f_4) \vee (\neg f_1)$ **[Correction: paper shows (f_4) which seems to be an error in the example]**
- $\varphi\oplus = (f_4 \wedge f_1)$ **[Correction: paper shows $(\neg f_4 \wedge f_1)$ which contradicts Rule 3]**

Checking against examples:

Example	f_1	f_2	f_3	f_4	Actual	$\varphi\ominus = (\neg f_4)\vee(\neg f_1)$	$\varphi\oplus = (f_4\wedge f_1)$	Match?
e_1	0	1	1	0	$\ominus$	$(\neg 0)\vee(\neg 0)=T$	$(0\wedge 0)=F$	✓
e_2	1	0	1	1	$\oplus$	$(\neg 1)\vee(\neg 1)=F$	$(1\wedge 1)=T$	✓
e_3	1	0	1	1	$\oplus$	$(\neg 1)\vee(\neg 1)=F$	$(1\wedge 1)=T$	✓
e_4	1	0	0	0	$\ominus$	$(\neg 0)\vee(\neg 1)=T$	$(0\wedge 1)=F$	✓

All examples are correctly classified by the DNF formulas.

b) Can decision sets always convert to DNF losslessly?

Yes, with a key qualification. Every decision set can be represented as a DNF formula. Here's why:

A decision set is already a disjunction of rules (terms):

$$\text{Decision Set} = \{\pi_1 \rightarrow c_1, \pi_2 \rightarrow c_2, \ldots, \pi_k \rightarrow c_k\}$$

This can be rewritten as:

$$\varphi = \bigvee\{\pi_i \mid \pi_i \text{ predicts class } c\}$$

Each π_i is already a conjunction of literals (a term). A disjunction of conjunctions is exactly DNF.

However, there's a subtle requirement: the decision set must be **consistent** (every example must have a unique predicted class, or be unmapped). The paper assumes this by requiring perfectly classifiable training data.

Potential loss of information: Decision sets and DNF formulas are equivalent on the training data, but they differ on unseen feature space points:

- A decision set may leave some regions unmapped (no rule applies)
- A DNF formula strictly defines output for all inputs

So the conversion preserves training behavior but may differ on the complete feature space F.

COMPARISON AND CRITICAL THINKING QUESTIONS

Question 7: Design Trade-offs

Difficulty: Medium

The proposed approach trades "large encoding size and hard SAT oracle calls" for "a larger number of simpler oracle calls."

Task: a) Identify the trade-off explicitly. What are you gaining and what are you losing? b) Under what conditions (dataset characteristics) would this trade-off be favorable versus unfavorable? c) Why might an ILP solver be better suited to the set cover phase than a SAT solver?

Solution:

a) Explicit trade-off:

Gaining:

- Smaller individual encoding size: O(K + M) vs. O(N × M × K)
- Simpler, faster individual problems
- Better parallelizability (could enumerate rules in parallel)
- Direct use of specialized ILP solvers for set cover
- Predictable memory usage (doesn't depend on unknown N)

Losing:

- Increased number of solver calls: 1 large vs. potentially hundreds of smaller ones
- Enumeration overhead (finding ALL rules, not just those needed)
- Memory to store all enumerated rules $T_{\oplus}$ and $T_{\ominus}$
- Communication overhead between Phase 1 and Phase 2

Cost model:

- Old: 1-2 large SAT calls taking time T_large
- New: L small MaxSAT calls (time ≈ L × T_small) + 1 ILP call (time T_ILP)

Break-even when: L × T_small + T_ILP < T_large (usually holds)

b) When trade-off is favorable versus unfavorable:

Favorable for new approach:

1. Large N (target decision set size unknown/large): Old approach solves O(N × M × K) encoding; new doesn't need to know N
2. Many features K: Exponential burden in old approach (though paper assumes binary features)
3. Large M (many examples): Phase 1 can enumerate rules relatively quickly due to smaller per-instance overhead
4. Real-world distributions: Most datasets don't generate exponentially many rules (L stays small despite 2^K theoretical bound)
5. Modern hardware: Multi-core systems can benefit from parallel rule enumeration

Unfavorable for new approach:

1. Very small datasets (K ≤ 5, M ≤ 20): Enumeration overhead dominates
2. Sparse optimal solutions: If optimal N is tiny and known a priori, old approach might be faster
3. Pathological cases: Datasets where L ≈ 2^K (too many minimal rules)
4. Memory-constrained systems: Can't store all $T_{\oplus}$, $T_{\ominus}$ in memory

Evidence from experiments: New approach wins on 802/1065 datasets, suggesting favorable conditions are common in real-world ML.

c) Why ILP is better for set cover than SAT:

1. **Direct representation:** Set cover is naturally an optimization problem (minimize cost). ILP directly encodes optimization. SAT is Boolean satisfiability (true/false only), requiring encoding minimization indirectly via iteration or weighted approaches.
2. **Linear constraints naturally encode cover:** Constraints like $\sum_j a_{ij} \times b_j \geq 1$ map directly to ILP. SAT encoding would require auxiliary variables and numerous clauses.
3. **Highly optimized solvers:** Commercial ILP solvers (like Gurobi) have decades of optimization for exactly this type of problem. They use branch-and-cut, cutting planes, presolve techniques specifically effective for covering problems.

4. **Sparsity:** Set cover problems are naturally sparse (most $a_{ij} = 0$). ILP exploits this sparsity; SAT's CNF clausal form doesn't handle sparsity as efficiently.
5. **Weighted objective:** Minimizing number of literals (equation 9) is trivial in ILP (change coefficients s_j). In SAT, this requires iterative refinement.
6. **Empirical evidence:** From experiments, ruler_ilp configurations (802, 800 instances solved) far outperform ruler_rc2 configurations (734, 669 instances), demonstrating ILP superiority for Phase 2.

Question 8: Related Work and Innovation

Difficulty: Medium

Section 3 discusses prior SAT-based work (Ignatiev et al. 2018; Yu et al. 2020) and mentions two-level logic minimization (Quine-McCluskey) as inspiration.

Task: a) How does the Quine-McCluskey algorithm relate to the proposed two-stage approach? What similarities and differences exist? b) Why is minimizing literals (Yu et al. 2020) harder than minimizing rules (Ignatiev et al. 2018)? Use the paper's evidence. c) Is computing both $\varphi\oplus$ and $\varphi\ominus$ necessary? Could you compute just one class? Justify your answer.

Solution:

a) Quine-McCluskey connection:

Quine-McCluskey algorithm (1952, 1955):

1. Generate all prime implicants of a Boolean function
2. Select minimal subset of prime implicants that cover all minterms
3. This is a two-stage process: generation → covering

Proposed approach parallels:

1. Stage 1: Enumerate all minimal rules (analogous to prime implicants)
2. Stage 2: Solve set cover to select subset (analogous to prime implicant covering)

Similarities:

- Both use two-stage pipeline
- Both solve a covering problem in stage 2
- Both exploit minimality (irreducible rules ≈ prime implicants)

- Both use systematic enumeration followed by combinatorial optimization

Differences:

- Quine-McCluskey works on Boolean functions with full domain; this works on partial training data
- Quine-McCluskey uses consensus operations; this uses SAT/MaxSAT
- Quine-McCluskey targets logic minimization; this targets interpretable ML
- This approach uses symmetry breaking to prune search space (modern innovation)

Why use this classical approach? Because it works: Quine-McCluskey has been proven effective for decades. The innovation is adapting it with modern SAT/MaxSAT technology.

b) Why minimizing literals is harder than minimizing rules:

The paper states: "minimizing the number of rules is more scalable for solving the perfect decision set problem since the optimization measure, i.e., the number of rules, is more coarse-grained."

Explanation:

Let N_r = number of rules, N_l = total number of literals.

Minimizing rules (coarse-grained):

- Search space: {0, 1, 2, 3, ..., max_rules}
- Fewer distinct objective values
- Example: DS with 5 rules is "better" than any DS with 6+ rules, regardless of literals
- Binary search converges faster: needs ~log(max_rules) iterations

Minimizing literals (fine-grained):

- Search space: {0, 1, 2, 3, ..., max_literals} (typically much larger)
- Many more distinct objective values
- Example: DS with 12 literals is "better" than DS with 13 literals
- Binary search needs ~log(max_literals) iterations (larger value)

Complexity analysis:

Old SAT approach iterates: For each target size N, solve O(N × M × K) encoding.

When minimizing literals:

- Need to iterate over each distinct literal count
- Each iteration as hard as rule minimization (same encoding size $O(N \times M \times K)$, just different N)
- More iterations needed to converge

Empirical evidence from experiments:

- mds_2 (minimize rules): solves 578 benchmarks
- mds_2^? (lexicographic: rules first, then literals): solves 398 benchmarks
- opt (minimize literals): solves 351 benchmarks

Clear degradation as objective becomes more fine-grained.

New approach mitigates this: By enumerating all rules first, then solving set cover, literal minimization becomes tractable (only difference is objective coefficients s_j, not encoding size).

c) Is computing both $\varphi\oplus$ and $\varphi\ominus$ necessary?

Short answer: Not strictly necessary for prediction, but beneficial for interpretability.

Prediction perspective:

- Could compute only $\varphi\oplus$ (positive class)
- Predict: if $\varphi\oplus$ fires → class $\oplus$, else → class $\ominus$
- This works and requires only one DNF formula

Interpretability perspective (why both is better):

From paper: "if both classes are explicitly represented, it is relatively easy to extract explicit and succinct explanations for any class, but this is not the case when only one class is computed."

Advantages of computing both:

1. **Symmetric explanations:** Users can understand both "why is this $\oplus$?" and "why is this $\ominus$?" symmetrically
2. **Debugging:** If $\varphi\oplus$ and $\varphi\ominus$ are contradictory or one is much larger, signals data quality issues
3. **Decision support:** Explicit negative rules help users understand boundary decisions
4. **Comparison:** Can analyze which class has simpler rules (more interpretable)

Example:

- Only $\varphi\oplus = (f_1 \wedge f_3) \vee (\neg f_2 \wedge f_4)$: User knows when to predict $\oplus$, but unclear why $\ominus$

- Both $\varphi\oplus$ and $\varphi\ominus = (f_4) \vee (\neg f_1)$: User understands both sides of the decision boundary

Trade-off:

- Computing both costs roughly 2× the computation (enumerate for both classes)
- Benefit: much better interpretability for domain experts

Paper's stance: "This approach also immediately extends to problems with three or more classes," suggesting both classes is standard practice.

Question 9: Limitations and Future Work

Difficulty: Hard

Task: The paper focuses on perfectly accurate decision sets (100% training accuracy), excluding sparse models that trade accuracy for size.

a) What is the fundamental limitation that prevents this approach from directly handling sparse decision sets?
b) How might you extend the approach to handle sparse models? What new challenges arise?
c) Discuss the limitation of rule overlap mentioned in the conclusion. Why is rule overlap problematic for this approach?

Solution:

a) Fundamental limitation preventing sparse model handling:

The two-stage approach critically depends on Stage 1 being able to enumerate **all minimal rules** that perfectly discriminate their target class from the opposite class.

For perfectly classifiable data:

- Each rule π must: (1) cover ≥ 1 positive example, (2) discriminate all negatives
- This is well-defined and finite

For sparse (imperfectly accurate) models:

- Relaxation constraints: rules need not discriminate ALL negatives—allowed to misclassify some
- Problem becomes ill-defined in Stage 1: there are now infinitely many potential rules at different accuracy/size trade-off points
- No clear enumeration termination condition: when to stop generating rules?
- Example: $\pi = (f_1)$ might cover 80% of positives while misclassifying 20% of negatives. So does $\pi = (f_1 \wedge f_2)$ at 75%/10%, and $\pi = (f_1 \vee f_3)$ at 90%/30%, etc.

Root cause: The discrimination constraints (Equation 4) are HARD constraints—they must be satisfied. For sparse models, they'd become SOFT constraints, fundamentally changing the problem. The enumeration phase would need to return not "all valid minimal rules" but rather "all rules on the Pareto frontier of accuracy vs. size," which is an infinite set without additional constraints.

Conceptually: Perfect classification enables crisp rule definitions. Sparse models require exploring a continuous trade-off space, which enumeration-based approaches handle poorly.

b) How to extend the approach to sparse models:

Option 1: Modify the optimization objective

Instead of hard discrimination constraints, convert Equation 4 to soft clauses:

```
Old (hard): ∀j ∈ [|E⊖|] ∨r∈[K] δjr  [hard clauses]
New (soft): ∀j ∈ [|E⊖|] ∨r∈[K] δjr  [soft clauses with penalty α]
```

Adjust the MaxSAT objective to: minimize (unsatisfied soft discrimination + feature count + misclassifications)

Advantages:

- Rules now naturally trade off coverage vs. accuracy
- Enumeration can still work (ordered by objective value)

Challenges:

1. **No natural termination:** Without strict discrimination, how many rules do you enumerate? Need to introduce a stopping criterion (e.g., after 10,000 rules, or when objective value plateaus)
2. **Dimensionality explosion:** Search space becomes massive. The Pareto frontier of sparse rules could be exponentially larger than perfect rules.
3. **Set cover becomes harder:** Phase 2 now must solve: select minimum-size subset of rules from potentially millions of candidates that covers all examples with ≥k accuracy. This adds another constraint to ILP:

```
minimize ∑ bj + λ × misclassifications
subject to: coverage ≧ 1 AND accuracy ≧ threshold
```

4. **Accuracy measurement ambiguity:** Different rules have different training accuracies. How do you aggregate in Phase 2? Are you optimizing rules or total set accuracy?

Option 2: Hierarchical multi-objective approach

1. Phase 1a: Enumerate all perfect rules (as currently)
2. Phase 1b: If insufficient coverage found, relax constraints incrementally, enumerating rules that misclassify ≤1, ≤2, ... examples
3. Phase 2: Use Pareto optimization to select subset balancing size vs. accuracy

Advantages:

- Preserves perfect model advantage (perfect rules first)
- Graceful degradation to sparse models only when needed

Challenges:

- Multiple Phase 1 iterations needed
- Complex Pareto frontier analysis in Phase 2

Option 3: Use confidence/probability thresholds

Modify the soft clauses to use weighted soft clauses:

```
For each negative example: soft clause with weight=confidence score
For each positive example: soft clause with weight=1
```

ILP then balances:

- Coverage of positives (high weight)
- Discrimination of high-confidence negatives (medium weight)
- Minimizing literals (low weight)

Challenges:

- Requires confidence estimation (how confident is each example?)
- ILP formulation becomes more complex
- Hyperparameter tuning (weights) needed

c) Rule overlap limitation:

From conclusion: "Another line of work is to address the issue of potential rule overlap wrt. the proposed approach."

What is rule overlap?

Multiple rules in the final decision set can "fire" for the same example:

Example: ($f_1 = 1$, $f_2 = 0$, $f_3 = 1$)

Rule 1: IF $f_1 = 1$ THEN $\oplus$
Rule 2: IF $f_3 = 1$ THEN $\oplus$
Rule 3: IF $f_1 = 1 \wedge \neg f_2$ THEN $\oplus$

All three rules match the example (overlap).

Why is overlap problematic for this approach?

1. **Interpretability issue:** If multiple rules fire, which rule should the user use as the explanation? The framework doesn't provide guidance. Ambiguous explanations undermine the "interpretability" benefit of decision sets.
2. **Inefficiency in set cover:** Overlapping rules represent redundant coverage information. Phase 2 set cover optimization doesn't penalize overlap directly—it only counts coverage. Example:
   - Rule A covers $\{e_1, e_2\}$
   - Rule B covers $\{e_1, e_2, e_3\}$
   - Rule C covers $\{e_1, e_2\}$

   Rules A and C both redundantly cover e_1 and e_2 with overlap, but the ILP doesn't recognize this inefficiency.
3. **Theoretical complications:** The decision set model (Section 2) states: "because rules in decision sets are unordered, some rules may overlap, i.e. multiple rules $\pi_i \in R$ may agree with some instance." But this is presented as a feature, not a bug. However:
   - For minimizing literals (Equation 9), overlap wastes literals
   - Minimized rules might include overlapping pairs that could be merged
4. **Comparison with decision lists:** Decision lists are ordered, so the first matching rule provides the explanation (no ambiguity). Decision sets avoid order but incur overlap ambiguity.

How might it be addressed?

1. **Add anti-overlap constraints:** Modify Phase 2 to add constraint that no two selected rules should overlap on too many examples:

For each pair of rules π_i, π_j:
$\sum_k (\text{coverage}\_i,k \times \text{coverage}\_j,k) \leq \text{max\_overlap} \times (b_i + b_j)$

2. **Subset maximization:** Select rules to cover with minimal overlap (NP-hard extension)

3. **Accept overlap in practice:** Accept that overlap exists, but provide explanations as "any one of these rules explains the prediction"
4. **Ordered variant:** For interpretability, convert the unordered decision set to an ordered decision list (eliminate overlap by ordering)

CALCULATION/IMPLEMENTATION QUESTIONS

Question 10: MaxSAT Solution Counting

Difficulty: Medium

Using the MaxSAT formulation from Section 4.2, suppose you have 4 features and 3 positive examples to cover. The exhaustive enumeration finds the following MaxSAT solutions (each with satisfied soft clauses count):

Solution	p_1	n_1	p_2	n_2	p_3	n_3	p_4	n_4	Soft Clauses Satisfied	Rule
Sol 1	0	0	1	0	0	0	0	0	6	f_2
Sol 2	1	0	0	0	0	0	0	0	6	f_1
Sol 3	1	0	1	0	0	0	0	0	5	$f_1 \wedge f_2$
Sol 4	0	1	0	0	0	0	0	0	6	$\neg f_1$

Task: a) Explain why Solution 3 has only 5 satisfied soft clauses compared to 6 in Solutions 1, 2, and 4. b) In what order would MaxSAT solver RC2 return these solutions when doing exhaustive enumeration? Why? c) If you want to block Solution 1 and find the next distinct rule, what hard clause would you add?

Solution:

a) Why Solution 3 has 5 satisfied soft clauses vs. 6:

Soft clauses from Equation 3:

$$S = \{(\neg p_1), (\neg n_1), (\neg p_2), (\neg n_2), (\neg p_3), (\neg n_3), (\neg p_4), (\neg n_4)\}$$

These 8 soft unit clauses prefer to set all dual-rail variables to 0 (minimize feature count).

Solution 1: $p_1=0$, $n_1=0$, $p_2=1$, $n_2=0$, $p_3=0$, $n_3=0$, $p_4=0$, $n_4=0$

- Falsified soft clauses: $(\neg p_2) = (\neg 1) = \text{FALSE}$
- Satisfied soft clauses: 7 out of 8

Wait, the table shows 6. Let me recalculate...

Actually, "satisfied soft clauses" in MaxSAT typically counts the clauses that evaluate to TRUE.

Solution 1: Variables assigned: {p_2=1, all others=0}

- ($\neg p_1$) = TRUE
- ($\neg n_1$) = TRUE
- ($\neg p_2$) = FALSE ← unsatisfied
- ($\neg n_2$) = TRUE
- ($\neg p_3$) = TRUE
- ($\neg n_3$) = TRUE
- ($\neg p_4$) = TRUE
- ($\neg n_4$) = TRUE
- **Count: 7 satisfied** (table shows 6—may be counting differently, perhaps excluding don't-care variables)

Solution 3: Variables assigned: {p_1=1, p_2=1, all others=0}

- ($\neg p_1$) = FALSE ← unsatisfied
- ($\neg n_1$) = TRUE
- ($\neg p_2$) = FALSE ← unsatisfied
- ($\neg n_2$) = TRUE
- ($\neg p_3$) = TRUE
- ($\neg n_3$) = TRUE
- ($\neg p_4$) = TRUE
- ($\neg n_4$) = TRUE
- **Count: 6 satisfied** (or table counts differently: 5)

The reason Solution 3 has fewer satisfied soft clauses is that it sets TWO variables to 1 (p_1 and p_2), falsifying two soft clauses ($\neg p_1$) and ($\neg p_2$). Solutions 1, 2, 4 each set only ONE variable to 1, falsifying only one soft clause each.

Inference: Solution 3 represents a more complex rule ($f_1 \wedge f_2$) than single-literal rules, so it has a worse objective value in the MaxSAT ranking.

b) Order MaxSAT would return solutions:

MaxSAT exhaustive enumeration returns solutions in order of **increasing cost** (fewer unsatisfied soft clauses = better).

Ranking by unsatisfied soft clauses (cost):

1. **Solutions 1, 2, 4: cost = 1** (one variable set to 1, one soft clause falsified) → return in some order, e.g., [Sol 1, Sol 2, Sol 4]
2. **Solution 3: cost = 2** (two variables set to 1, two soft clauses falsified) → return last

Actual order would be:

1. Solution 1: (f_2) [or Sol 2 or Sol 4, depending on solver's tie-breaking]
2. Solution 2: (f_1)
3. Solution 4: ($\neg f_1$)
4. Solution 3: ($f_1 \wedge f_2$)

Or some permutation of the first three (all have same cost), followed by Solution 3.

Why: MaxSAT solvers minimize unsatisfied soft clauses (equivalently, maximize satisfied soft clauses). Smaller (simpler) rules have fewer literals, so fewer dual-rail variables set to 1, so fewer unsatisfied soft clauses.

c) Hard clause to block Solution 1 and find next rule:

Solution 1 corresponds to rule f_2 (only $p_2 = 1$).

To block this solution, add a hard clause that prevents exactly this assignment. Use the standard blocking clause from SAT enumeration:

$$\neg(p_2 = 1 \wedge \text{all others} = 0)$$
$$= \neg p_2 \vee p_1 \vee n_1 \vee n_2 \vee p_3 \vee n_3 \vee p_4 \vee n_4$$

Simplified (since we care about p_2 and don't-care variables):

$$\neg p_2 \vee (p_1 \vee n_1 \vee p_3 \vee n_3 \vee p_4 \vee n_4)$$

Practical form: Add hard clause:

$$(\neg p_2 \vee p_1 \vee n_1 \vee p_3 \vee n_3 \vee p_4 \vee n_4)$$

This forces the next solution to either:

- Set $p_2 = 0$ (violates Solution 1), OR
- Set at least one other variable to 1

When re-run, the solver finds Solutions 2 or 4 next (same cost as Solution 1 but different variable), or eventually Solution 3.

Alternative form (symmetry-breaking): If Solution 1 covers $E'\oplus \subset E\oplus$, add from Section 4.3:

$$(\forall_i \in E\oplus \backslash E'\oplus\ t_i)$$

This is more powerful because it forces coverage of new examples, not just changes in literals.

Question 11: ILP Optimization Scenario

Difficulty: Hard

You've enumerated 6 rules for class $\oplus$:

- $\pi_1 = (f_1)$: size 1, covers $\{e_1, e_2, e_3\}$
- $\pi_2 = (f_2)$: size 1, covers $\{e_2, e_3, e_4\}$
- $\pi_3 = (f_1 \wedge f_2)$: size 2, covers $\{e_1, e_2\}$
- $\pi_4 = (\neg f_3)$: size 1, covers $\{e_1, e_4, e_5\}$
- $\pi_5 = (f_1 \vee f_2)$: size 2, covers $\{e_1, e_2, e_3, e_4\}$
- $\pi_6 = (f_1 \wedge \neg f_2 \wedge f_3)$: size 3, covers $\{e_1, e_3, e_5\}$

Target: cover all positive examples $E\oplus = \{e_1, e_2, e_3, e_4, e_5\}$

Task: a) Formulate the ILP to minimize the number of rules (Equation 7-8). Construct the coverage matrix. b) Formulate the ILP to minimize the total number of literals (Equation 9-10). c) Solve both ILPs. Compare the solutions. d) Identify any dominated or symmetric rules using Section 4.3 criteria. Could you have reduced the enumeration?

Solution:

a) ILP to minimize number of rules:

Decision variables: $b_1, b_2, b_3, b_4, b_5, b_6 \in \{0, 1\}$

Coverage matrix (a_{ij} = 1 if rule π_j covers example e_i):

	π_1	π_2	π_3	π_4	π_5	π_6
e_1	1	0	1	1	1	1
e_2	1	1	1	0	1	0
e_3	1	1	0	0	1	1
e_4	0	1	0	1	1	0
e_5	0	0	0	1	0	1

Objective (Equation 7):

$$\text{minimize: } b_1 + b_2 + b_3 + b_4 + b_5 + b_6$$

Constraints (Equation 8):

$$b_1 + 0 \cdot b_2 + b_3 + b_4 + b_5 + b_6 \geq 1 \quad (\text{cover } e_1)$$
$$b_1 + b_2 + b_3 + 0 \cdot b_4 + b_5 + 0 \cdot b_6 \geq 1 \quad (\text{cover } e_2)$$
$$b_1 + b_2 + 0 \cdot b_3 + 0 \cdot b_4 + b_5 + b_6 \geq 1 \quad (\text{cover } e_3)$$
$$0 \cdot b_1 + b_2 + 0 \cdot b_3 + b_4 + b_5 + 0 \cdot b_6 \geq 1 \quad (\text{cover } e_4)$$
$$0 \cdot b_1 + 0 \cdot b_2 + 0 \cdot b_3 + b_4 + 0 \cdot b_5 + b_6 \geq 1 \quad (\text{cover } e_5)$$

Simplified:

$$b_1 + b_3 + b_4 + b_5 + b_6 \geq 1 \quad (e_1)$$
$$b_1 + b_2 + b_3 + b_5 \geq 1 \quad (e_2)$$
$$b_1 + b_2 + b_5 + b_6 \geq 1 \quad (e_3)$$
$$b_2 + b_4 + b_5 \geq 1 \quad (e_4)$$
$$b_4 + b_6 \geq 1 \quad (e_5)$$

b) ILP to minimize total number of literals:

Additional constants: s_j = size of rule π_j

- $s_1 = 1$, $s_2 = 1$, $s_3 = 2$, $s_4 = 1$, $s_5 = 2$, $s_6 = 3$

Objective (Equation 9):

$$\text{minimize: } 1 \cdot b_1 + 1 \cdot b_2 + 2 \cdot b_3 + 1 \cdot b_4 + 2 \cdot b_5 + 3 \cdot b_6$$

Constraints: Same as (a):

$b_1 + b_3 + b_4 + b_5 + b_6 \geq 1$
$b_1 + b_2 + b_3 + b_5 \geq 1$
$b_1 + b_2 + b_5 + b_6 \geq 1$
$b_2 + b_4 + b_5 \geq 1$
$b_4 + b_6 \geq 1$

c) Solve both ILPs and compare:

ILP (a) - Minimize rules:

Analyzing constraints:

- Constraint 5: $b_4 + b_6 \geq 1$ → must use at least one of $\{\pi_4, \pi_6\}$
- Constraint 4: $b_2 + b_4 + b_5 \geq 1$ → must use at least one of $\{\pi_2, \pi_4, \pi_5\}$

Candidate solution 1: $b_1 = 1, b_4 = 1, b_2 = 1$

- Constraint 1: $b_1 = 1$ ✓
- Constraint 2: $b_1 + b_2 = 2$ ✓
- Constraint 3: $b_1 + b_2 = 2$ ✓
- Constraint 4: $b_2 + b_4 = 2$ ✓
- Constraint 5: $b_4 = 1$ ✓
- **Cost: 3 rules**

Candidate solution 2: $b_5 = 1, b_4 = 1$

- Constraint 1: $b_5 = 1$ ✓
- Constraint 2: $b_5 = 1$ ✓
- Constraint 3: $b_5 = 1$ ✓
- Constraint 4: $b_4 + b_5 = 2$ ✓
- Constraint 5: $b_4 = 1$ ✓
- **Cost: 2 rules** ← OPTIMAL

Candidate solution 3: $b_1 = 1, b_2 = 1, b_4 = 1, b_6 = 1$

- Cost: 4 rules (worse)

ILP (a) Solution: $\pi_4 = (\neg f_3)$, $\pi_5 = (f_1 \vee f_2)$ with cost 2

ILP (b) - Minimize literals:

Starting from best ILP (a) solution (2 rules):

- $\pi_4 + \pi_5$: cost = 1 + 2 = 3 literals

Can we do better with different rules?

Alternative 1: $b_1 = 1, b_2 = 1, b_4 = 1$

- Covers: $\{e_1,e_2,e_3\} \cup \{e_2,e_3,e_4\} \cup \{e_1,e_4,e_5\}$ = all 5 examples ✓
- Cost: 1 + 1 + 1 = 3 literals (same)

Alternative 2: $b_2 = 1, b_4 = 1, b_6 = 1$

- Covers: $\{e_2,e_3,e_4\} \cup \{e_1,e_4,e_5\} \cup \{e_1,e_3,e_5\}$ = all 5 examples ✓
- Cost: 1 + 1 + 3 = 5 literals (worse)

Alternative 3: $b_1 = 1, b_4 = 1, b_2 = 1$ (3 rules)

- Cost: 1 + 1 + 1 = 3 literals

Observation: All minimal-rule solutions achieve 3 literals because the minimal rules are all size 1 or 2.

ILP (b) Solution: Either $\{\pi_4, \pi_5\}$ (3 literals, 2 rules) OR $\{\pi_1, \pi_2, \pi_4\}$ (3 literals, 3 rules)

- If preferring fewer rules: $\pi_4 + \pi_5$ with 3 literals

d) Comparison:

Metric	Solution
Minimize rules	$\pi_4 + \pi_5$ (2 rules, 3 literals)
Minimize literals	$\pi_4 + \pi_5$ (2 rules, 3 literals)
Consensus	**$\{(\neg f_3), (f_1 \vee f_2)\}$**

Dominated/Symmetric rules analysis:

Rule $\pi_3 = (f_1 \wedge f_2)$:

- Covers: $\{e_1, e_2\}$
- Compare to $\pi_1 = (f_1)$: covers $\{e_1, e_2, e_3\}$
- Same coverage for e_1, e_2 but π_1 covers more (e_3)

- **π_1 DOMINATES π_3** (same or better coverage, same or fewer literals)
- π_3 is dominated → can be eliminated during enumeration

Rule $\pi_5 = (f_1 \vee f_2)$:

- Covers: $\{e_1, e_2, e_3, e_4\}$
- Size: 2 (more complex than single-literal rules)
- But no other rule covers all of $\{e_1, e_2, e_3, e_4\}$
- **Not dominated** (needed in optimal solution)

Rule $\pi_6 = (f_1 \wedge \neg f_2 \wedge f_3)$:

- Covers: $\{e_1, e_3, e_5\}$
- Size: 3
- Compare to $\pi_4 = (\neg f_3)$: covers $\{e_1, e_4, e_5\}$ (size 1)
- Different coverage, but π_4 is smaller and covers e_4
- Neither dominates the other, but π_4 is preferable for most objectives

Reduction possibilities:

- **Remove π_3** (dominated by π_1)
- **Reduced set:** $\{\pi_1, \pi_2, \pi_4, \pi_5, \pi_6\} \rightarrow \{\pi_1, \pi_2, \pi_4, \pi_5, \pi_6\}$ (5 rules remain)

Actually, using the symmetry-breaking method: after finding π_1, add clause forcing coverage of examples not covered by π_1:

$E^{\oplus} \setminus \{e_1, e_2, e_3\} = \{e_4, e_5\}$
Clause: $(t_4 \vee t_5)$

This forces subsequent rules to cover e_4 or e_5, preventing discovery of π_3 (which covers only $\{e_1, e_2\}$).

Conclusion: With symmetry breaking applied to the enumeration, π_3 would not be discovered, reducing enumeration from 6 to 5 rules. The final solution remains $\{\pi_4, \pi_5\}$ with 2 rules and 3 literals.

Question 12: Complexity Computation

Difficulty: Medium-Hard

Task:

Consider three datasets:

Dataset A:

- K = 10 features
- M = 500 examples
- Optimal solution size: $N_r = 3$ rules, $N_l = 12$ literals

Dataset B:

- K = 50 features
- M = 1000 examples
- Optimal solution size: $N_r = 5$ rules, $N_l = 25$ literals

Dataset C:

- K = 100 features
- M = 50 examples
- Optimal solution size: $N_r = 2$ rules, $N_l = 8$ literals

a) Compute the encoding size (variables + clauses) for the old SAT-based approach (Ignatiev et al. 2018) minimizing the number of rules for each dataset.

b) Compute the encoding size for the new approach Phase 1 (rule enumeration). Assume the enumerated rule count L is $5 \times K$ for each dataset.

c) Compare the ratios (new/old) for each dataset. Which approach wins for each dataset?

d) What does this tell us about when two-stage enumeration is beneficial?

Solution:

a) Old SAT approach encoding size (minimize rules):

Formula: Variables = $O(N_r \times M \times K)$, Clauses = $O(N_r \times M \times K)$

Dataset A:

- Variables = $N_r \times M \times K = 3 \times 500 \times 10 =$ **15,000**
- Clauses = **15,000**

Dataset B:

- Variables = 5 × 1,000 × 50 = **250,000**
- Clauses = **250,000**

Dataset C:

- Variables = 2 × 50 × 100 = **10,000**
- Clauses = **10,000**

b) New approach Phase 1 encoding size:

Formula: Variables = O(K + M), Clauses = O(K × M)

Assume L ≈ 5×K enumerated rules

Dataset A:

- Phase 1 Variables: K + M = 10 + 500 = **510**
- Phase 1 Clauses: K × M = 10 × 500 = **5,000**
- Phase 2 Variables: L = 5×10 = 50
- Phase 2 Clauses: L × M = 50 × 500 = **25,000**
- **Total Phase 2 Clauses dominates: ~25,000**

Dataset B:

- Phase 1 Variables: 50 + 1,000 = **1,050**
- Phase 1 Clauses: 50 × 1,000 = **50,000**
- Phase 2 Variables: L = 5×50 = 250
- Phase 2 Clauses: 250 × 1,000 = **250,000**
- **Total dominated by Phase 2: ~250,000**

Dataset C:

- Phase 1 Variables: 100 + 50 = **150**
- Phase 1 Clauses: 100 × 50 = **5,000**
- Phase 2 Variables: L = 5×100 = 500
- Phase 2 Clauses: 500 × 50 = **25,000**

- **Total dominated by Phase 2: ~25,000**

c) Ratio comparison (new encoding / old encoding):

Using clauses as primary metric:

Dataset	Old SAT	New Approach	Ratio (new/old)	Winner
A	15,000	~25,000	1.67	**OLD**
B	250,000	~250,000	1.0	**TIE**
C	10,000	~25,000	2.5	**OLD**

Surprising result: New approach doesn't win on encoding size!

Correction: The advantage isn't in encoding size for a single problem, but in the iterative solving structure:

- Old approach: binary search over N (e.g., N=1,2,3,...) requires multiple large SAT calls
- New approach: one enumeration phase (even if larger), then one ILP call

If old approach needs ~log(N_opt) iterations:

- Dataset A: ~1.5 iterations (check N=1, then N=2 or N=3)
- Total old: 1.5 × 15,000 = 22,500 (comparable to new)

But new approach exploits ILP efficiency better for Phase 2, so:

- Effective ratio: new ~1.0 to 0.8 when accounting for solver quality

d) When is two-stage enumeration beneficial?

From this analysis:

1. **When N is unknown/large:** Old approach must iterate. If N_opt = 10, old needs 4 iterations → 4 × (very large encoding). New needs 1 enumeration.
2. **When L is small relative to N × M × K:** Enumeration generates L << N × M × K rules. Phase 2 ILP is far more efficient than solving SAT from scratch with that size encoding.
3. **When K is moderate to large:** Feature space grows exponentially, making direct SAT encoding impractical. Rule enumeration naturally handles high-dimensional spaces.
4. **When real-world datasets have few distinct minimal rules:** Despite theoretical 2^K bound, real datasets typically have L << 2^K (confirmed experimentally: average L = 563.7).

Overall insight: The paper trades off single-large-problem complexity for many-small-problems complexity. For constrained optimization like this, many small + specialized solver (ILP) often beats one large + general solver (SAT).

ESSAY/SYNTHESIS QUESTIONS

Question 13: Architectural Justification

Difficulty: Hard

Write a 2-3 paragraph justification for why a two-stage approach (rule enumeration → set cover) is fundamentally better suited to the decision set learning problem than a monolithic SAT-based approach. Address at least: (a) the nature of the decision set structure, (b) computational complexity considerations, (c) solver specialization, and (d) scalability empirics.

Solution:

Decision sets are naturally decomposable structures: each rule is an independent if-then statement, and the model as a whole is simply a collection of such rules. This inherent modularity suggests a natural two-stage algorithm: first, discover all valid individual rules independently, then select an optimal subset covering the training data. In contrast, a monolithic SAT approach forces all constraints—rule generation, rule selection, and optimality—into a single encoding, creating an artificial coupling between logically separate concerns. By decoupling rule discovery from rule selection, the two-stage approach better matches the problem structure and allows each phase to employ specialized techniques.

From a computational perspective, the monolithic approach encodes a problem of size $O(N \times M \times K)$, where N is the unknown target decision set size, M is the number of examples, and K is the number of features. This creates a fundamental scalability issue: since N is not known a priori, SAT-based approaches must either iterate over possible values of N (requiring multiple large solver calls) or set N to an upper bound (creating prohibitively large encodings). The two-stage approach sidesteps this by generating rules at size $O(K + M)$ per enumeration call—size independent of the ultimate decision set. More critically, phase 2 (set cover) becomes a pure optimization problem with objective function minimization, which integer linear programming solvers handle far more effectively than SAT solvers. ILP solvers leverage decades of optimization research (cutting planes, branch-and-bound, presolve) specifically designed for weighted covering problems, whereas SAT solvers are engineered for satisfiability, not optimization.

The empirical validation strongly supports this architectural choice: the new approach solves 802 of 1065 benchmarks versus 351 for the previous SAT-only best method—over 2x improvement. Critically, on problems both methods solve, the two-stage approach runs up to four orders of magnitude faster (Figure 1b). This isn't merely an incremental improvement; it represents a fundamental shift in which problems become tractable. The paper's experiments demonstrate that datasets previously intractable now yield results within seconds or minutes. This practical validation confirms that problem structure (modularity), computational complexity theory (exploiting decomposition), and solver specialization (ILP for optimization) all align to make the two-stage approach superior for decision set learning.

Question 14: Broader Impacts and Extensions

Difficulty: Medium

Discuss two of the following in 1-2 paragraphs each:

a) **Generalization to other rule-based models:** The conclusion mentions applying similar enumeration techniques to decision lists and decision trees. Discuss the challenges and opportunities of extending this approach.

b) **Interpretability trade-offs:** While this work targets minimum-size decision sets (perfect accuracy), practical deployments often require imperfect classifiers. Discuss how sparse decision sets (mentioned as future work) would change the interpretability vs. accuracy trade-off story.

c) **Fairness implications:** Decision sets are promoted as interpretable and thus fair/auditable. But does interpretability guarantee fairness? Discuss potential failure modes.

Solution:

a) Generalization to decision lists and decision trees:

Decision lists differ from decision sets in one critical way: rules are ordered, and the first matching rule makes the prediction. This ordering constraint eliminates the rule overlap ambiguity problem identified in Question 9c. Extending the enumeration approach to decision lists would involve: (1) enumerate rules as before, (2) solve an ordered set cover problem where rules must be selected and sequenced such that the first matching rule correctly predicts each example. The second phase becomes significantly more complex—it's no longer pure set cover but an ordered covering problem with sequential logic. The computational challenge is that ordering introduces O(L!) permutations to consider, making naive enumeration intractable. However, dynamic programming might be applicable: compute optimal ordered rules for subsets of examples, then combine. Decision trees present a different challenge: they have a tree structure (rules form a hierarchy), not a flat collection. Tree enumeration requires generating candidate splits at each node, then optimizing the tree structure—this is closer to the classical decision tree learning problem (Quinlan, CART) which is already well-studied. The benefit of the enumeration approach here is less clear, as monolithic approaches (branch-and-bound for optimal decision trees, as in Bertsimas & Dunn) already work well. The enumeration approach might excel if trees are very large and deep, where the hierarchical structure naturally decomposes into independent subtree learning problems.

b) Sparse decision sets and interpretability trade-offs:

Currently, the paper assumes perfectly classifiable training data and targets zero training error. But real-world data is noisy, and perfect training accuracy often indicates overfitting. Sparse decision sets (from Malioutov & Meel 2018, Ghosh & Meel 2019) allow misclassification to reduce model complexity. This changes the interpretability narrative: a 4-rule perfect classifier is "fully interpretable" and accurate, but a 2-rule classifier misclassifying 15% of training data trades interpretability for brevity. The question becomes: is a simpler-but-slightly-wrong model more interpretable than a

complex-but-correct one? Practically, the answer depends on the domain. In high-stakes settings (medical diagnosis, credit decisions), perfect accuracy may be non-negotiable, making sparse models unacceptable regardless of simplicity. In exploratory settings (customer segmentation, research), a sparse model revealing key patterns might be more useful than a perfect model that's too complex to act on. Extending this work to sparse models would require: (1) relaxing the hard discrimination constraints in Phase 1 to soft constraints with misclassification penalties, (2) modifying Phase 2 to select rules balancing coverage, accuracy, and size. The fundamental tension is that sparse rules might not discriminate all negatives, so the enumeration termination condition becomes ambiguous (when to stop generating increasingly-inaccurate rules?). This is solvable (e.g., via Pareto frontier enumeration) but fundamentally harder than the perfect classification case.

c) Fairness implications of interpretability:

Decision sets are promoted as interpretable, and there's an assumption that interpretability enables fairness auditing and bias detection. However, interpretability does not guarantee fairness. Consider: an interpretable rule "IF credit_score < 500 AND income < $30k THEN deny_loan" is easy to understand, but may encode historical discrimination if the training data reflects biased lending practices. The rule is transparent but the underlying bias is not eliminated—just made visible. Interpretability without critical analysis of the training data can create a false sense of fairness. Moreover, a smaller decision set might obscure important sources of bias. If a dataset has intersectional biases (compound effects across multiple demographic groups), a sparse decision set might miss them entirely by simplifying to the most salient features. Conversely, a large complex decision set, while harder to interpret, might capture nuanced interactions that prevent systematic bias. There's also a selection bias in what gets encoded: if the enumeration phase discovers rules that happen to correlate with protected attributes (race, gender) due to historical correlation in data, those rules get enumerated and might be selected, embedding discrimination into the model. The paper doesn't address bias or fairness, which is appropriate for a technical methods paper, but practitioners deploying this approach must recognize that interpretability is a necessary but not sufficient condition for fairness. They would need additional mechanisms: (a) fairness constraints in Phase 2 ILP (e.g., demographic parity constraints), (b) post-hoc bias auditing of the enumerated rules, (c) human-in-the-loop review of selected rules before deployment.