Supplement: Satisfiability Solving with LLMs

ANONYMOUS AUTHOR(S)

ACM Reference Format:
Anonymous Author(s). 2018. Supplement: Satisfiability Solving with LLMs. *Proc. ACM Program. Lang.* 1, CONF, Article 1 (January 2018), 13 pages.

1 Mathematical Properties of ADR

In this section, we provide a rigorous derivation of the mathematical properties of the *Accurate Differentiation Rate (ADR)*. We show its relationship to conventional metrics such as accuracy, recall, and precision, establish tight upper and lower bounds, and argue why ADR serves as a stricter and more reliable measure of reasoning ability.

1.1 Setup and Notation

Consider M paired instances $\{(F_i^{\text{SAT}}, F_i^{\text{UNSAT}})\}_{i=1}^{M}$, each consisting of:

- one satisfiable formula F_i^{SAT},
- one unsatisfiable formula F_i^{UNSAT}.

For each pair, define binary correctness indicators:

$$A_i = \mathbf{1}\{\hat{y}(F_i^{\text{SAT}}) = \text{SAT}\}, \qquad B_i = \mathbf{1}\{\hat{y}(F_i^{\text{UNSAT}}) = \text{UNSAT}\}.$$

Then:

$$r_S = \frac{1}{M}\sum_{i=1}^{M} A_i, \quad r_U = \frac{1}{M}\sum_{i=1}^{M} B_i, \quad \text{ADR} = \frac{1}{M}\sum_{i=1}^{M} A_i B_i.$$

The per-instance accuracy over all $2M$ examples is:

$$\text{Acc} = \frac{1}{2M}\sum_{i=1}^{M}(A_i + B_i).$$

Lemma 1.1 (Accuracy identity).

$$\text{Acc} = \tfrac{1}{2}(r_S + r_U)$$

1.2 Upper Bounds of ADR

Theorem 1.2 (ADR bounded by recalls). *For any paired dataset,*

$$\text{ADR} \le \min(r_S, r_U)$$

Proof. For each pair, $A_iB_i \le A_i$ and $A_iB_i \le B_i$. Summing and normalizing proves the result.

Theorem 1.3 (ADR bounded by accuracy).

$$\text{ADR} \le \text{Acc}$$

Proof. For each pair, $A_iB_i \le \frac{1}{2}(A_i + B_i)$. Summing gives ADR $\le$ Acc.

2018. ACM 2475-1421/2018/1-ART1
https://doi.org/

1.3 Lower Bound of ADR

Lemma 1.4 (ADR lower bound).

$$\mathrm{ADR} \geq r_S + r_U - 1.$$

Equivalently,

$$\mathrm{ADR} \geq \max(0, r_S + r_U - 1)$$

Corollary 1.5 (ADR vs. accuracy). *Since* $\mathrm{Acc} = \frac{1}{2}(r_S + r_U)$,

$$\mathrm{ADR} \geq \max(0, 2\mathrm{Acc} - 1)$$

1.4 Independence Approximation

Theorem 1.6 (ADR under independence). *Let* $A = \{F^{SAT}$ *correct*$\}$ *and* $B = \{F^{UNSAT}$ *correct*$\}$. *Then*

$$\mathrm{ADR} = P(A \cap B) = r_S r_U + \mathrm{Cov}(\mathbf{1}_A, \mathbf{1}_B).$$

In particular, if A and B are independent,

$$\mathrm{ADR} = r_S r_U$$

Proof. By linearity of expectation,

$$\frac{1}{M} \sum A_i B_i = E[\mathbf{1}_A \mathbf{1}_B] = P(A)P(B) + \mathrm{Cov}(\mathbf{1}_A, \mathbf{1}_B).$$

If A and B are independent, the covariance vanishes.

1.5 Summary of Bounds

Theorem 1.7 (ADR inequality chain). *For any paired dataset,*

$$\max(0, r_S + r_U - 1) \;\leq\; \mathrm{ADR} \;\leq\; \min(r_S, r_U) \;\leq\; \mathrm{Acc} = \tfrac{1}{2}(r_S + r_U)$$

This chain shows that ADR is a stricter measure than accuracy: it requires *both* sides of each pair to be correct, rather than only one. Consequently, ADR resists dataset imbalance and prediction bias, and more faithfully captures true reasoning competence.

1.6 ADR vs. Traditional Metrics: Essential Differences

- **Accuracy (per instance).** Strongly affected by class distribution [Manning 2008; Powers 2020]; even if one side is always right and the other always wrong, accuracy can still be 0.5, which looks "not too bad." **Fails to reflect pairwise discrimination ability.**
- **Precision/Recall/F1 (require positive class).** Once SAT is defined as the positive class, in SAT-dominated regions the metric can be inflated by "always predicting SAT." [Powers 2020] **Completely blind to the UNSAT side.**
- **ADR (per pair).** Independent of distribution, forces both sides to be correct simultaneously; **directly measures whether the model can distinguish the pair.**

In one sentence: **Accuracy/F1 mean "average good enough," while ADR means "both sides of the same pair must be correct."**

1.7 Toy Counterexamples (More Intuitive)

(1) Always Predict SAT.. $r_S = 1, r_U = 0$. Then $\mathrm{Acc} = 0.5$, but $\mathrm{ADR} = 0$. Accuracy looks decent, but ADR pinpoints the failure: cannot distinguish paired differences.

(2) Strong on One Side, Moderate on the Other. $r_S = 1.0$, $r_U = 0.6$. Then ADR ≤ 0.6, and the lower bound $1.0 + 0.6 - 1 = 0.6$, so ADR $= 0.6$. Meanwhile, Acc $= \frac{1.0+0.6}{2} = 0.8$. **Conclusion:** Accuracy $= 0.8$ looks "good," but ADR reveals the real bottleneck of pairwise correctness is only 0.6.

(3) Anti-Calibrated (Always Flip Within Pair). Every pair "gets exactly one side correct" $\Rightarrow$ Acc $= 0.5$, but ADR $= 0$. ADR correctly identifies total failure to capture decisive edits.

1.8 Practical Insights

- **Why more suitable for our task?** This paper emphasizes the ability to distinguish *minimal structural edits.* ADR directly checks whether the SAT/UNSAT pair is simultaneously classified correctly, naturally avoiding α-driven class imbalance bias.
- **Connection to witness validity.** In experiments, models with higher ADR also tend to produce more often *valid satisfying assignments or constructions [Biere et al. 2009]* (e.g., packing/cover solutions). This shows ADR is closer to "structural reasoning" rather than label-guessing.
- **When to prefer ADR?** Whenever the evaluation target is pairwise/counterfactual discrimination, or when dataset distribution drifts with conditions (e.g., α), ADR should be prioritized. Accuracy/F1 can be reported as supplementary but cannot stand alone as evidence of reasoning.

1.9 One-Sentence Summary

ADR is a **pairwise, symmetric, distribution-independent** metric that requires models to get both SAT and UNSAT of the same pair correct. It effectively suppresses bias and opportunism, highlighting true **structural discrimination and reasoning ability**. Mathematically, ADR is upper-bounded by $\min(r_S, r_U)$ and Acc, and lower-bounded by $\max(0, r_S+r_U-1)$. These inequalities explain why ADR is stricter than Accuracy/F1 and better reveals the model's true reasoning capacity.

2 ADR vs. MCC: A Mathematical Comparison

In this section we rigorously compare the *Accurate Differentiation Rate (ADR)* with the *Matthews Correlation Coefficient (MCC)* in the paired-evaluation setting. We derive their closed-form relationship, express MCC in terms of pair-level decomposition, and highlight why ADR provides a clearer and more reliable measure of pairwise reasoning ability.

2.1 Pair Decomposition

Let M be the number of SAT/UNSAT pairs. For each $i = 1, \ldots, M$:

$$A_i = \mathbf{1}\{\hat{y}(F_i^{\mathrm{SAT}}) = \mathrm{SAT}\}, \quad B_i = \mathbf{1}\{\hat{y}(F_i^{\mathrm{UNSAT}}) = \mathrm{UNSAT}\}.$$

Define the pair-level counts:

$$\begin{aligned} n_{11} &= \#\{i : A_i = 1, B_i = 1\}, && \text{(both correct)} \\ n_{10} &= \#\{i : A_i = 1, B_i = 0\}, && \text{(SAT-only correct)} \\ n_{01} &= \#\{i : A_i = 0, B_i = 1\}, && \text{(UNSAT-only correct)} \\ n_{00} &= \#\{i : A_i = 0, B_i = 0\}, && \text{(both wrong).} \end{aligned}$$

Thus $M = n_{00} + n_{01} + n_{10} + n_{11}$, and

$$\mathrm{ADR} = \frac{n_{11}}{M}, \quad r_S = \frac{n_{10} + n_{11}}{M}, \quad r_U = \frac{n_{01} + n_{11}}{M}, \quad \mathrm{Acc} = \tfrac{1}{2}(r_S + r_U).$$

2.2 Closed-Form Expression of MCC

Theorem 2.1 (MCC in terms of recalls [Matthews 1975]). *On the balanced paired dataset, MCC can be expressed solely in terms of the recalls r_S and r_U:*

$$\mathrm{MCC} = \frac{r_S + r_U - 1}{\sqrt{(r_S + 1 - r_U)(r_U + 1 - r_S)}}.$$

Proof. Substituting

$$\mathrm{TP} = Mr_S, \quad \mathrm{TN} = Mr_U, \quad \mathrm{FP} = M(1 - r_U), \quad \mathrm{FN} = M(1 - r_S)$$

into the MCC definition and simplifying yields the stated result.

Corollary 2.2 (Undefined cases of MCC [Chicco and Jurman 2020]). *MCC is undefined if and only if $(r_S, r_U) \in \{(1,0),(0,1)\}$, i.e. the model always predicts one class. In contrast, ADR is always defined and equals 0 in these cases.*

2.3 Pair-Structured Expression

Lemma 2.3 (MCC in terms of ADR, β, and δ). *Define*

$$\delta = \tfrac{n_{10} - n_{01}}{M}, \qquad \beta = \tfrac{n_{00}}{M}.$$

Then

$$\mathrm{MCC} = \frac{\mathrm{ADR} - \beta}{\sqrt{1 - \delta^2}}. \tag{1}$$

Proof. Note that

$$r_S + r_U - 1 = \tfrac{n_{11} - n_{00}}{M} = \mathrm{ADR} - \beta,$$

and

$$r_S + 1 - r_U = 1 + \delta, \quad r_U + 1 - r_S = 1 - \delta.$$

Substituting into Theorem 2.1 proves the claim.

Corollary 2.4 (Sign condition). *From Lemma 2.3, $\mathrm{MCC} > 0$ if and only if $\mathrm{ADR} > \beta$, i.e. the proportion of both-correct pairs exceeds that of both-wrong pairs.*

2.4 Interpretation and Example

Equation (1) shows that MCC conflates joint correctness (ADR) with both-wrong frequency (β) and class asymmetry (δ). ADR, by contrast, isolates the joint-correct rate n_{11}/M, independent of δ.

Example. Let $r_S = r_U = 0.8$. Then Theorem 2.1 gives $\mathrm{MCC} = 0.6$ regardless of joint distribution. But ADR satisfies

$$0.6 = \max(0, r_S + r_U - 1) \;\le\; \mathrm{ADR} \;\le\; \min(r_S, r_U) = 0.8.$$

Thus two models can share the same MCC (0.6) yet have ADR = 0.6 (errors all one-sided) or ADR = 0.8 (errors all joint), revealing different discrimination abilities.

2.5 Summary

Theorem 2.5 (ADR vs. MCC). *On the balanced paired dataset,*

$$\mathrm{MCC} = \frac{\mathrm{ADR} - \beta}{\sqrt{1 - \delta^2}}, \quad \beta = \tfrac{n_{00}}{M}, \quad \delta = \tfrac{n_{10} - n_{01}}{M}.$$

Hence:

(1) ADR is always defined and directly measures joint correctness.

(2) *MCC is undefined under class-collapse and mixes ADR with β and δ.*
(3) *Two models with identical MCC can differ in ADR, making ADR a stricter probe of pairwise reasoning.*

3 VERTEX COVER, EQUIVALENT OR NOT EQUIVALENT?

3.1 From CNF to Graph: Motivation and Hypothesis

A central claim of this paper, reiterated in Section **??**, is that *SAT solving is a foundational reasoning capability for LLMs*. Because a wide range of tasks in AI, program analysis, and combinatorial optimization reduce to SAT, an LLM's performance on SAT is an informative proxy for its potential to generalize across diverse reasoning problems. At the same time, SAT itself admits standard polynomial-time reductions to other canonical problems—notably *Vertex Cover*. If LLMs genuinely capture the *logical structure* underlying SAT, then their behavior should be *representation-invariant*: given an instance expressed as a CNF formula or as its graph-theoretic reduction, the model ought to reach consistent conclusions.

Representation-Invariance Question. Experiment 4 is designed to test precisely this question: ***RQ6:*** *Do LLMs reason about satisfiability in a manner that is stable across equivalent representations, or are apparent successes tied to the surface form of CNF?* Concretely, we take the paired SAT/UNSAT formulas from Experiment 3 in Section **??** and apply a standard reduction from CNF-SAT to Vertex Cover, producing graph instances (V, E) with a bound k such that the formula is satisfiable iff the constructed graph admits a vertex cover of size $\leq k$. We then prompt LLMs on the graph representation to decide the existence of such a cover and, when answering YES, to output a concrete vertex set.

Evaluation Plan. This setup allows us to assess two aspects of reasoning: (i) *task performance* on graphs (Accuracy/Precision/Recall/F1), and (ii) *cross-representation consistency* between CNF and graph inputs, i.e., whether categorical decisions (SAT vs. UNSAT vs. YES vs. NO) and, when applicable, the structural content of the prediction (returned cover sets) align across the two representations. High consistency would support the view that LLMs are leveraging common logical invariants rather than superficial input patterns; conversely, a sharp drop when moving from CNF to graphs would suggest representation- sensitive heuristics rather than genuine reasoning.

The remainder of this section presents the prompting protocol, datasets, and metrics, followed by results and analysis.

3.2 Reduction from CNF to Graph

It is well known that propositional formulas in conjunctive normal form (CNF) can be reduced to graph-based formulations of classical traditional NP-complete problems. In particular, CNF-SAT can be polynomially reduced to the *Vertex Cover* problem, which asks whether there exists a set of vertices of size at most k that covers all edges in a given undirected graph. Specifically, a CNF formula can be transformed into a directed acyclic graph (DAG) or equivalently into an undirected bipartite graph, where clauses and literals are represented as nodes and edges capture their relationships. If a CNF formula is satisfiable, then there exists a corresponding vertex cover of bounded size in the constructed graph. This reduction provides a graph-theoretic perspective on satisfiability and allows SAT solving to be studied through the lens of combinatorial optimization.

3.3 Experiment 5 setup.

Building on the paired SAT/UNSAT formulas from Experiment 3, we applied standard reductions to generate their corresponding graph instances for the vertex cover problem. Each graph was then provided to the LLMs with a structured prompt designed to query for a vertex cover of size at most k. An example of the prompt template is shown below:

```python
def make_vc_prompt_core(V, E, k):
    V_str = "{ " + ", ".join(str(v) for v in V) + " }"
    E_str = "{ " + ", ".join("{" + f"{u}, {v}" + "}" for (u, v) in E) + " }"
    return (
      "You are solving a Vertex Cover decision task.\n"
      f"Graph: V={V_str}, E={E_str}\n"
      f"Question: Is there a vertex cover C with |C| <= {k}?\n\n"
      "OUTPUT (one-line JSON only):\n"
      '  {"answer":"YES"|"NO","cover":[...],"explain":"..."}\n'
      "Rules:\n"
      f"  - Return YES only if you can list C (|C|<= {k}) that covers every edge.\
    n"
      "  - Otherwise return NO and set cover=[].\n"
      "  - No extra text outside the JSON.\n"
    )
```

Listing 1. Core prompt for Vertex Cover (abbreviated). This listing shows only the decision-critical instructions and output schema. See Appendix 5 for the full prompt (edge-case rules, formatting guards) and the deterministic checker.

Each instance was given in the form (V, E) with vertex set V, edge set E, and the bound k. The model was required to output a JSON object specifying whether a vertex cover of size $\leq k$ exists, and, if so, to list the corresponding cover set.

3.4 Evaluation on Graph Inputs

Traditional metrics on graphs (Figure 1 (a–d)). Across Accuracy, Precision, Recall, and F1, most models operate near the random-guessing regime when asked to decide whether a small graph admits a vertex cover of size $\leq k$. Two notable exceptions are `GPT-5` and, to a lesser extent, `deepseek-reasoner`:

- **Most models:** Accurcay curves remain clustered around ≈ 0.5 and degrade with N, indicating little signal beyond chance.
- **`GPT-5`:** achieves near-perfect scores for small instances (N=5, 8, 10), but collapses as N grows, mirroring the scalability trend observed on CNF inputs.
- **`deepseek-reasoner`:** performs better than chance at small N, yet still trails `GPT-5` and loses reliability as instance size increases.

These outcomes show that simply re-expressing CNF as graphs does not make the decision task easier for current LLMs; genuine gains are model-dependent and fragile under scaling.

3.5 Cross-Representation Consistency

Prediction alignment across CNF and graphs (Figure 1 (e): vc_cnf_equivalence). Despite weak absolute accuracy for many models on graphs, their *decisions* are largely *consistent* across equivalent representations: for most models, the fraction of instances where the CNF prediction matches the Vertex-Cover prediction exceeds **80%**. This means that—even when they are not correct—models tend to preserve a stable decision rule when the SAT instance is recast as a graph, suggesting sensitivity to structure beyond the surface form of CNF.

Agreement with ground truth across both views (Figure 1 (f): vc_cnf_label_equivalence). Requiring *triple agreement* (CNF decision = graph decision = true label) reveals the boundary of this transfer.

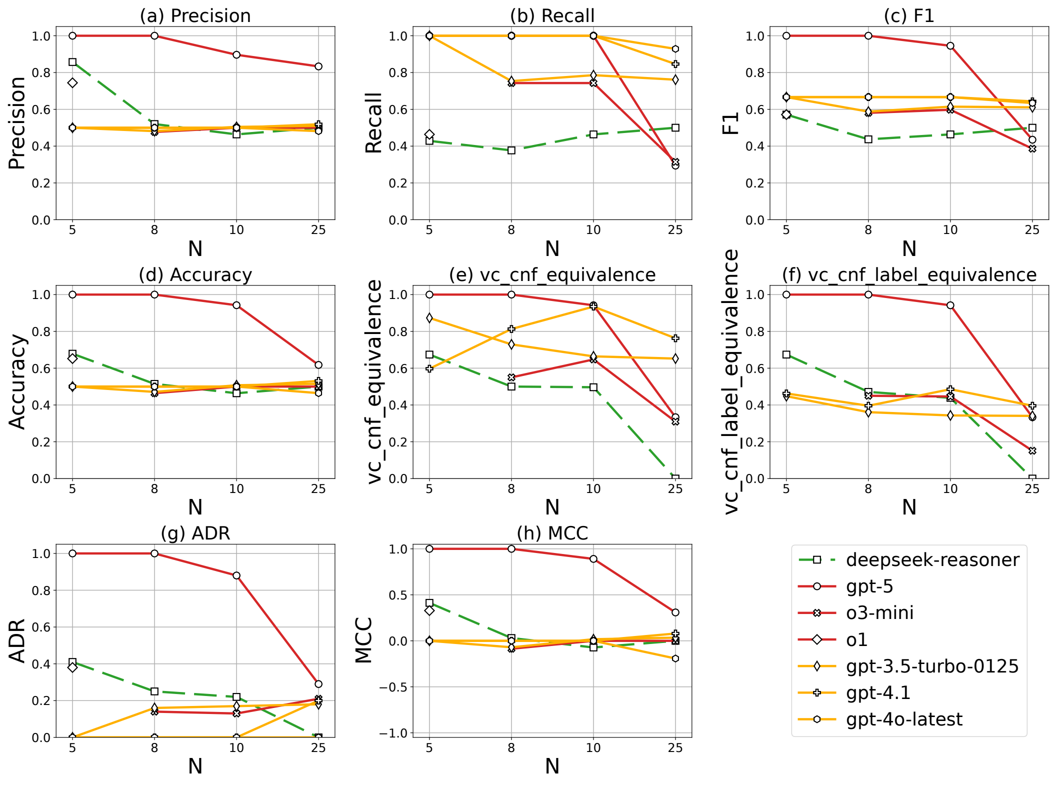

Fig. 1. Vertex Cover results across models. **(e)** *vc_cnf_equivalence*: CNF–graph decision agreement; typically > 80% for most models, indicating a stable decision rule across representations (independent of correctness). **(f)** *vc_cnf_label_equivalence*: triple agreement among CNF decision, graph decision, and ground truth; reflects true correctness.

Most models remain near chance, indicating they are *consistently wrong across representations.* **Crucially, `GPT-5` is the exception:** its CNF and graph decisions are not only highly consistent but also *correct* on small instances, yielding clearly non-random triple agreement. This provides direct evidence for representation-invariant *and* truth-aligned reasoning—exactly the behavior our research question seeks to detect.

Answer to the Research Question. LLMs *do* exhibit representation-invariant behavior: CNF and graph decisions largely agree (Figure 1(e)). For most models, the observed invariance primarily reflects adherence to a stable heuristic applied across representations. Notably, `GPT-5` exhibits representation invariance that is both consistent and accurate, answering our research question affirmatively: satisfiability judgments can remain stable across equivalent representations while aligning with the ground truth.

Finally, Figure 1(g) and (h) compare ADR and MCC under the paired evaluation scheme. As in prior sections, ADR isolates the true pair-level differentiation ability, while MCC is confounded by systematic biases. The degradation of both metrics with growing N reaffirms scalability as the dominant bottleneck across models.

3.6 Implications

- **SAT as a foundational problem.** The high CNF–graph decision alignment (Figure 1 (e)) shows that the logical structure captured by SAT transfers to Vertex Cover. This supports our thesis from Section ??: SAT encapsulates core invariants that underlie other reasoning tasks.
- **Invariance is useful only with base competence.** Figure 1 (f) makes clear that representation stability alone is insufficient: without strong base competence, models remain consistently wrong across representations. `GPT-5` stands out because it achieves both high invariance and correctness on small N, demonstrating genuine structure-sensitive reasoning rather than surface heuristics.
- **Scalability remains the bottleneck.** Even for `GPT-5`, performance degrades as N increases, echoing the trend on CNF inputs. To leverage representation-invariant reasoning reliably, models must improve scale robustness.

Most LLMs preserve their decision boundary when moving from CNF to graphs, but only `GPT-5` (on small instances) preserves a *correct* boundary. Thus, SAT indeed functions as a *foundational* problem: success on SAT not only predicts transfer to other reductions (e.g., Vertex Cover) but also reveals whether a model's invariance reflects genuine logical competence or merely stable yet incorrect heuristics.

4 3SAT Transfer to 3D Packing

4.1 Motivation.

As outlined in Section ??, 3SAT serves as a foundational probe of LLM reasoning. To test *problem-family transfer* beyond CNF and graphs, we map 3SAT instances into a discrete *3D packing* formulation, which represents another NP problem class, such that the feasibility of the packing instance directly corresponds to the satisfiability of the original formula. If models truly capture the underlying logical structure, their behavior on packing should echo their behavior on 3SAT, just as we observed for Vertex Cover.

4.2 Reduction and prompt.

Each 3SAT instance is reduced to a grid container with two object types and hard constraints: (i) per x-index, select exactly one of two fixed-orientation rods (at $y = 0$ or $y = 1$) spanning the full depth; (ii) for each depth layer z, place exactly one unit "token" at a coordinate from its `allowed_slots`; and (iii) a token is valid only if covered by the chosen rod at the same (x, y). Feasibility requires that all tokens can be placed under these rules. We use a structured, one-line-JSON prompt that enforces format and validity checks:

```python
def make_3d_packing_prompt(instance, instance_name=None):

    X, Y, Z = instance["container"]["size"]
    header = (
    "You are an expert in discrete 3D packing and constraint solving.\n"
    f"- Container (grid): X = {X}, Y = {Y}, Z = {Z}; rotations are NOT allowed and
     orientations are locked.\n"
    "- Objects come in TWO types (no other types exist):\n"
    "  * Type-A (long rods): for each x-index in {1..X} there are EXACTLY TWO rods
    , anchored at y=0 and y=1, "
    "each with size [1,1,Z] and spanning cells (x=const, y in {0,1}, z=0..Z-1). "
    "You must SELECT EXACTLY ONE rod per x-index (choose y=0 OR y=1), never both.\
    n"
    '  * Type-B (unit cubes, "tokens"): there is EXACTLY ONE Type-B object for each depth z=j (j=0..Z-1). '
    "Each Type-B object lists its allowed placements as coordinates in its `allowed_slots` field.\n"
    "- Valid placement rule:\n"
    "  * A Type-B placement at (x,y,z=j) is VALID ONLY if that cell is covered by a SELECTED Type-A rod at the same (x,y).\n"
    "  * You MUST choose coordinates ONLY from the `allowed_slots` provided; do not invent coordinates.\n"
    "- Feasibility means EVERY Type-B object can be placed validly under these rules.\n\n"
    "Return ONLY a ONE-LINE JSON object:\n"
    '  {"answer":"YES"|"NO",'
    ' "assignment":{"1":true/false,"2":true/false,...,"' + str(X) + '":true/false },'
    ' "tokens":[[x,y,z],...],'
    ' "explain":"<= 2 sentences"}\n'
    "Where:\n"
    '- `assignment` uses STRING KEYS for ALL x-indices 1..X; true => select the rod at y=0, false => select the rod at y=1.\n'
    "  - `tokens` must contain EXACTLY one coordinate for EACH depth layer z=0..Z-1, in ascending z order, "
    "and each coordinate must be one of that layer's `allowed_slots`.\n"
    "If ANY Type-B object cannot be validated, return NO with empty assignment and tokens.\n"
    "Do not include any extra text before or after the one-line JSON.\n"
    )

    return header
```

Listing 2. Prompt builder for 3D Packing

4.3 Example 3-SAT → 3D-packing (two clauses)

Consider the 3-CNF

$$\Phi \;=\; (x_1 \vee \neg x_2 \vee x_3) \;\wedge\; (\neg x_1 \vee x_2 \vee x_3).$$

There are $n = 3$ variables and $m = 2$ clauses. We construct the packing instance as follows.

Container (how the size is derived). We use a discrete grid of size $[X, Y, Z] = [n, 2, m] = [3, 2, 2]$:

- $X = n$ indexes variables along the x-axis ($x = 1, 2, 3$).
- $Y = 2$ encodes Boolean truth as layers: $y = 0$ means True, $y = 1$ means False.
- $Z = m$ indexes clauses along depth: $z = 0, 1$.

Type-A objects (rods): size and position, and where they come from. For each variable x_i we create two rods, one for True and one for False. Each rod spans *all* clause layers so that any clause token placed at depth z can be covered by the chosen truth-layer for that variable.

- **Size.** Every rod has size $[1, 1, m] = [1, 1, 2]$.
- **Anchors (positions).** Rods are axis-aligned and anchored at $z = 0$:

$$\text{True-rod of } x_i: \;(x = i,\; y = 0,\; z = 0), \qquad \text{False-rod of } x_i: \;(x = i,\; y = 1,\; z = 0).$$

- **Constraint.** *Exactly one* rod must be selected per variable (choose either the True-rod or the False-rod).

For this example ($i \in \{1, 2, 3\}$) the six rods are:

$$\begin{aligned}&\text{rod\_T\_x1 at } (1, 0, 0),\ \text{rod\_F\_x1 at } (1, 1, 0),\\ &\text{rod\_T\_x2 at } (2, 0, 0),\ \text{rod\_F\_x2 at } (2, 1, 0),\\ &\text{rod\_T\_x3 at } (3, 0, 0),\ \text{rod\_F\_x3 at } (3, 1, 0),\end{aligned}$$

each with size $[1, 1, 2]$ and extending over $z = 0, 1$.

Type-B objects (tokens): size, allowed slots, and how they are derived from clauses. We create one token per clause and restrict where it may be placed using the clause's literals.

- **Size.** Every token has size $[1, 1, 1]$ (occupies a single grid cell).
- **Allowed-slot rule.** For clause C_j ($j \in \{0, 1\}$):

$$x_i \implies (x = i,\ y = 0,\ z = j), \qquad \neg x_i \implies (x = i,\ y = 1,\ z = j).$$

- **Allowed slots for this example.**
  - $C_0 = (x_1 \lor \neg x_2 \lor x_3)$ at $z = 0$ gives

$$\{(1, 0, 0),\ (2, 1, 0),\ (3, 0, 0)\}.$$

  - $C_1 = (\neg x_1 \lor x_2 \lor x_3)$ at $z = 1$ gives

$$\{(1, 1, 1),\ (2, 0, 1),\ (3, 0, 1)\}.$$

Feasibility rule (semantic constraint). A packing is feasible iff *every* clause token is placed on one of its allowed slots *and* the chosen cell is covered by the rod selected for that variable at the same (x, y). Equivalently, feasibility of the packing is equivalent to satisfiability of Φ.

A SAT assignment and the corresponding feasible packing. The formula is satisfiable. For example, take

$$x_1 = \text{True}, \quad x_2 = \text{True}, \quad x_3 = \text{True}.$$

- **Selected rods (from the assignment).** Choose the True-layer rods

$$(x, y) = (1, 0),\ (2, 0),\ (3, 0),$$

  i.e., `rod_T_x1`, `rod_T_x2`, `rod_T_x3`, each of size $[1, 1, 2]$.
- **Token placements (one per clause depth).**
  - For C_0 at $z = 0$, pick $(1, 0, 0)$ (allowed by literal x_1 and covered by `rod_T_x1`).
  - For C_1 at $z = 1$, pick $(2, 0, 1)$ (allowed by literal x_2 and covered by `rod_T_x2`).

Both tokens use allowed slots and lie on selected rods, so the packing is feasible, matching the fact that Φ is SAT.

4.4 Experimental setup.

We convert the **70 paired** 3SAT instances (SAT/UNSAT) used in our prior experiment 3 in Section ?? into 3D-packing instances and query the same set of LLMs with Listing 2. Models produce a categorical feasibility decision and, when answering `YES`, a concrete rod selection and token placements.

4.5 Analysis of Results

Figure 2 summarizes the 3D–packing results under the one-line–JSON protocol when the model must output a binary feasibility decision and, on "YES", a concrete assignment. The trends closely mirror those observed for CNF and Vertex Cover.

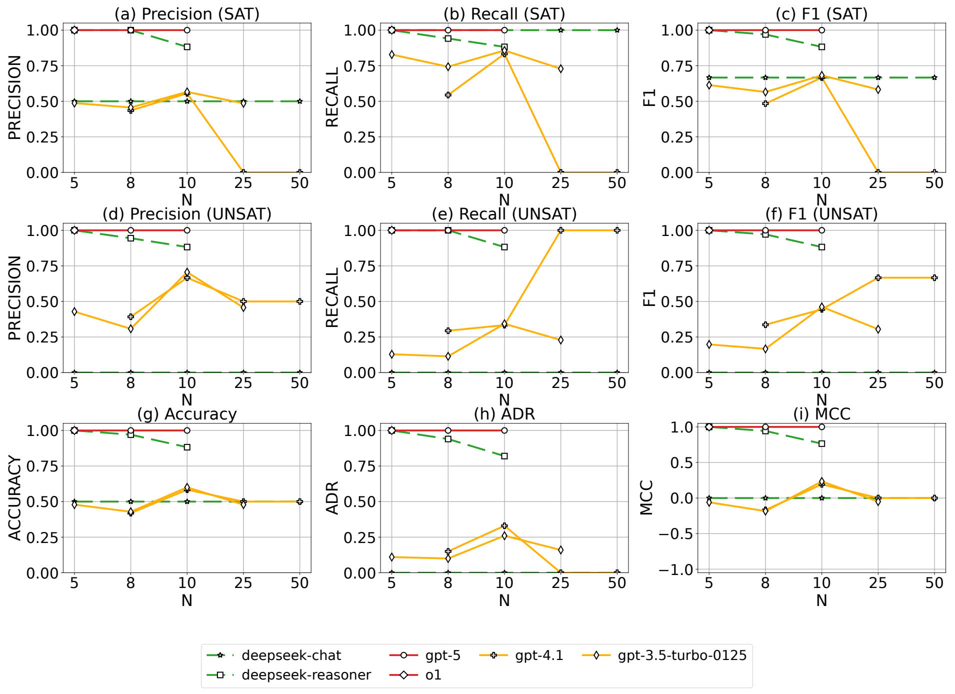

Fig. 2. 3D packing: predictions when LLM answers "yes" (3×3).

(i) Conventional metrics look strong at small N but collapse with scale. On the SAT side (panels (a)–(c)), *reasoning-oriented* models (e.g., `gpt-5`, `deepseek-reasoner`) attain near-perfect precision/recall/F1 for $N \leq 10$, then degrade sharply as N increases; non-reasoning baselines (`deepseek-chat`, etc.) hover around a mid-level plateau and never approach solver-like performance. On the UNSAT side (panels (d)–(f)), we see the usual asymmetry: some models gain recall at larger N by predicting NO more often, but this comes with low precision, yielding unstable F1. These patterns reproduce the class-skew sensitivities discussed in Section **??**.

(ii) Accuracy and MCC are poor diagnostics; ADR is the most informative. Overall accuracy (panel (g)) remains near chance for several models even when one class is systematically mishandled, confirming that accuracy alone hides failure modes. MCC (panel (i)) is positive for small N but quickly drifts toward 0 as N grows; as analyzed in Section 2, MCC mixes joint correctness with class imbalance and can look acceptable while pairwise discrimination is failing. By contrast, *ADR* (panel (h)) gives a clean signal: reasoning models exhibit high ADR at small N (i.e., they simultaneously get both members of a SAT/UNSAT pair right), followed by a monotone decline toward chance with increasing N. This agrees with the mathematical properties in Section 1 and with the CNF/VC findings in Section **??** and Section 3.

(iii) Problem-family transfer and representation invariance. The scaling profile in packing echoes 3SAT and Vertex Cover: strong small-N behavior for reasoning models, rapid convergence to chance at larger N, and persistent SAT bias for non-reasoning models. This confirms *problem-family transfer*: models port both their strengths and their weaknesses from CNF to packing. Moreover, high ADR coincides with higher *Packing-Assignment Validity* (when "YES" is returned), indicating that

pairwise discrimination aligns with producing logically valid witnesses—a representation-robust sign of structural reasoning.

4.6 Implications

The packing formulation preserves the essential difficulty landscape of 3SAT. Traditional metrics can overstate competence due to class imbalance and prediction bias, whereas ADR (together with witness checking) faithfully tracks reasoning capacity and its scale-induced degradation. These results reinforce our thesis that **SAT is a foundational probe**: success (or failure) on SAT predicts transfer to other NP-style reductions such as 3D packing.

5 Appendix: Packing and Vertex Cover Prompts

5.1 Full Vertex Cover Prompt Template

Listing 3 shows the complete version of the Vertex Cover prompt. Unlike the abbreviated form in the main text (Listing 1), here we retain all formatting guards, edge-case instructions, and multiple valid output examples.

```python
def make_vc_prompt(V, E, k, instance_name=None):
    """
    V: list (sorted)
    E: list of 2-tuples (sorted, endpoints sorted)
    k: int
    return: prompt string
    """
    V_str = "{ " + ", ".join(str(v) for v in V) + " }"
    E_str = "{ " + ", ".join("{" + f"{u}, {v}" + "}" for (u, v) in E) + " }"

    header = (
        "You are an expert on the graph vertex cover problem. "
        "The following is an undirected graph represented as a pair where "
        "the first component is the set of vertices and the second is a set of edges. "
        f"Given this graph, determine whether there exists a vertex cover of size <= {k}.\n\n"
        "IMPORTANT OUTPUT FORMAT:\n"
        "Return ONLY a single-line JSON object with keys:\n"
        '  - "answer": either "YES" or "NO"\n'
        '  - "cover": a list of distinct integers (the vertex set) if and only if "answer" is "YES"; otherwise []\n'
        '  - "explain": a short plain-text explanation (max 1-2 sentences)\n'
        "Rules:\n"
        f"  * Output \"YES\" ONLY IF you can provide a concrete vertex set C with |C| <= {k} that covers EVERY edge in E.\n"
        "  * If you are unsure OR cannot verify ALL edges are covered OR cannot list such C, output \"NO\" and set cover=[].\n"
        "  * No text outside the JSON. One line only.\n"
        "Example valid outputs:\n"
        '{"answer":"YES","cover":[1,3,7],"explain":"All edges incident to at least one of 1,3,7."}\n'
        f'{{"answer":"NO","cover":[],"explain":"Unable to verify a cover within size {k}."}}\n'
    )

    name_line = (f"Instance: {instance_name}\n" if instance_name else "")
    body = (
        f"{name_line}"
        "Graph (V, E):\n"
        f"V = {V_str}\n"
        f"E = {E_str}\n"
        f"k = {k}\n\n"
        "Task: Decide if there exists a vertex cover of size <= k.\n"
        "If and only if YES, return a valid cover set and a brief explanation in
    the JSON as described."
    )
    return header + "\n" + body
```

Listing 3. Full prompt builder for Vertex Cover instances (full version).

6 Conclusion

Need to add a conclusion.

Data Availability

References

Armin Biere, Marijn Heule, and Hans van Maaren. 2009. *Handbook of satisfiability*. Vol. 185. IOS press.

Davide Chicco and Giuseppe Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics* 21, 1 (2020), 6.

Christopher D Manning. 2008. *Introduction to information retrieval.* Syngress Publishing,.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405, 2 (1975), 442–451.

David MW Powers. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061* (2020).