Explaining SAT Solving Using Causal Reasoning

Introduction

- Boolean Satisfiability (SAT) is a fundamental problem in computer science that involves determining whether there exists an assignment σ that satisfies a given Boolean formula
- The modern CDCL SAT solvers owe their performance to well-designed and tightly integrated core components: branching phase selection, clause learning , restarts, and learnt clause cleaning

Research Gap

- Traditional complexity-theoretic studies focus on analyzing the limitations of SAT solvers
- SAT solvers struggle for certain instances
- Hypotheses are mostly derived from the researcher or developer’s intuition and empirically tested in an ad-hoc manner
- Some research done such as:
  - certain branching heuristics work particularly well in combination with specific restart heuristics
  - the decay factor used in activity calculations must be adjusted according to the active restart heuristics

Research Goals

- what factors influence the utility of clauses in memory management in SAT solvers
- causal model, a graph whose nodes represent different components and heuristics of the solver, and its edges represent causal relations between them
- applying a set of principled rules known as do-calculus to compute the effect of a particular factor on the utility of clauses
- inquire about some unresolved questions related to the solving process,

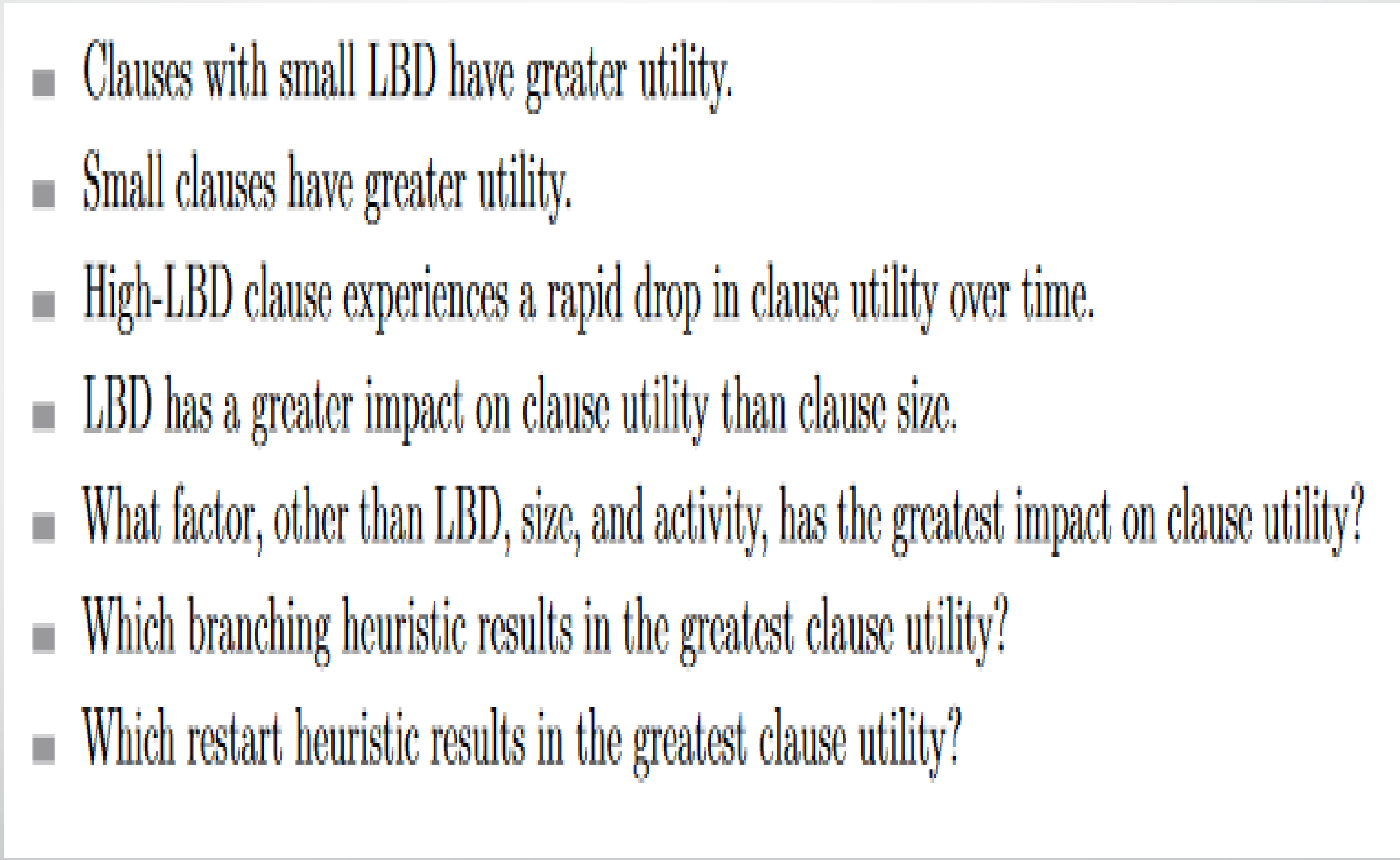
CCDL Solver

- CDCL-based SAT solvers begin with an initially empty set of assignments
- solver incrementally assigns a subset of variables until the current partial assignment is unable to satisfy the current formula
- solver employs a backtracking mechanism to trace the cause of unsatisfiability
- may run for millions of conflicts during a single execution
- periodically remove some of the less useful clauses to maintain solver efficiency

Learnt Claude Learning

- these heuristics evaluate the “utility” of a clause based on some parameters and determine whether to retain or discard it
- Een employed the notion of activity (or VSIDS) as a surrogate for utility
- Audemard utilized literal block distance (LBD) as a proxy for utility
- However both were unable to prove the proper reasoning behind the decision and had some unanswered questions

Questions about current SAT solvers



- Clauses with small LBD have greater utility.
- Small clauses have greater utility.
- High-LBD clause experiences a rapid drop in clause utility over time.
- LBD has a greater impact on clause utility than clause size.
- What factor, other than LBD, size, and activity, has the greatest impact on clause utility?
- Which branching heuristic results in the greatest clause utility?
- Which restart heuristic results in the greatest clause utility?

Casual Model

- the set of features that cause an outcome and the function that measures the effect of the features on the outcome
- he underlying representation of the causal model is (1) a directed acyclic graph (DAG) $G = (V, E)$ where the vertices V are the set of random variables $V = \{X1, . . . , Xn\}$ and the edges $(Xi, Xj) \in E$ represent causal relationships
- causal graph has a quantitative interpretation: we use it to answer do-queries that estimate the causal effect between a treatment variable and an outcome variable.
- do-query: We say there is a causal effect between a treatment variable Xi and an outcome variable Y if under an intervention on the treatment variable Xi, the outcome variable changes
- The do operator thus changes the value of Xi while keeping every other variable in V the same, except for those directly or indirectly affected by Xi
- Average Treatment Effect (ATE). Given a causal graph, a treatment variable X, and an outcome variable Y , the average treatment effect measures the change of the expected value of the outcome variable when we intervene on the treatment variable and change it from a constant value b to a
- (Average Treatment Effect). The average treatment effect of a variable X (called the treatment) on the target variable Y (called the outcome) is: $ATE(X, Y, a, b) = E [Y |do(X = a)] – E [Y |do(X = b)]$,
- (Conditional Average Treatment Effect). The conditional average treatment effect of X on Y conditioned on a variable of interest W is: $CATE(X, Y, W, a, b) = E [Y |do(X = a), W] – E [Y |do(X = b), W]$

Structure

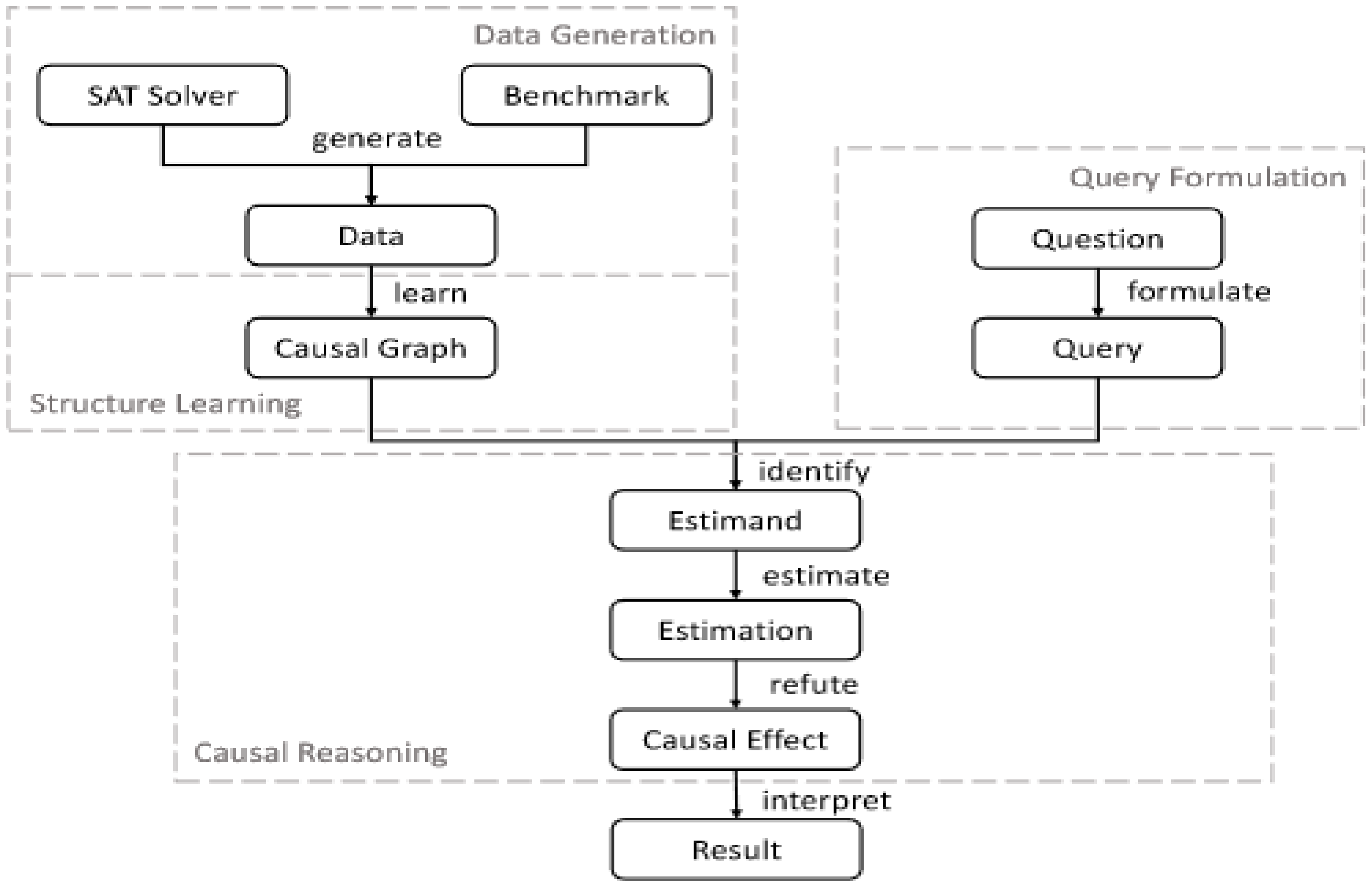
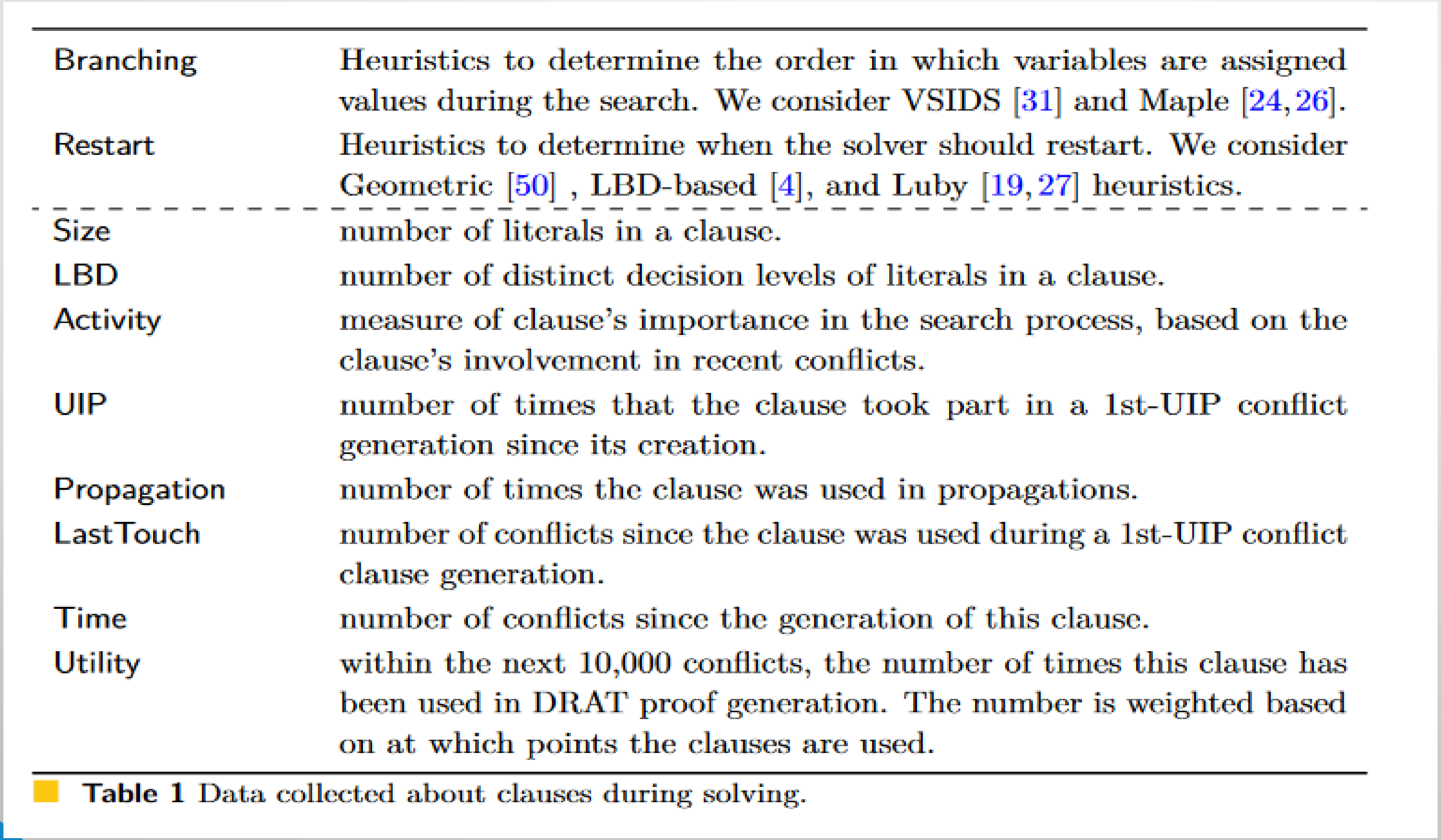


Figure 2 Our approach overview, from data generation to the causal estimate.

Features

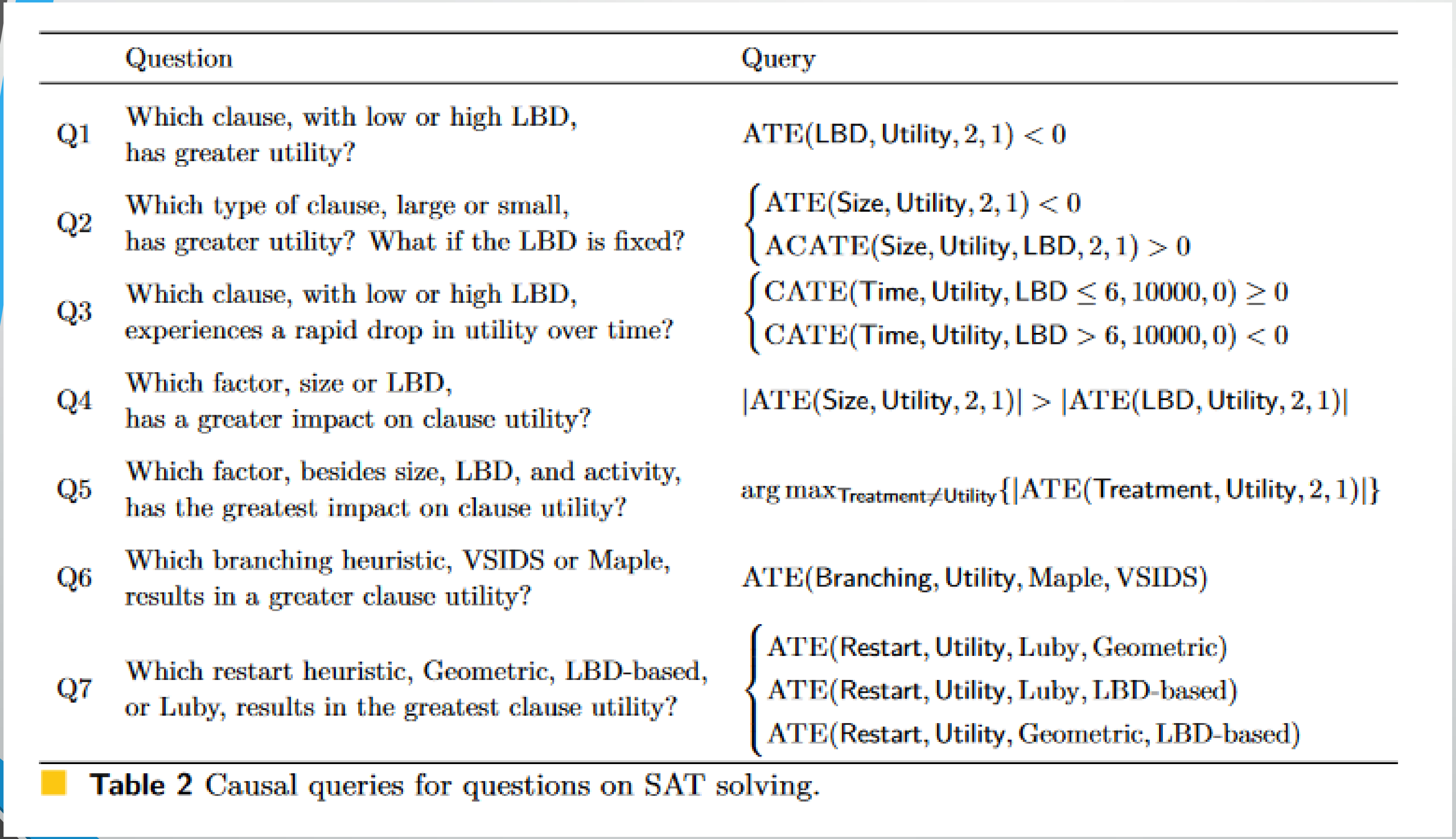


Branching	Heuristics to determine the order in which variables are assigned values during the search. We consider VSIDS [31] and Maple [24, 26].
Restart	Heuristics to determine when the solver should restart. We consider Geometric [50] , LBD-based [4], and Luby [19, 27] heuristics.
Size	number of literals in a clause.
LBD	number of distinct decision levels of literals in a clause.
Activity	measure of clause's importance in the search process, based on the clause's involvement in recent conflicts.
UIP	number of times that the clause took part in a 1st-UIP conflict generation since its creation.
Propagation	number of times the clause was used in propagations.
LastTouch	number of conflicts since the clause was used during a 1st-UIP conflict clause generation.
Time	number of conflicts since the generation of this clause.
Utility	within the next 10,000 conflicts, the number of times this clause has been used in DRAT proof generation. The number is weighted based on at which points the clauses are used.

Table 1 Data collected about clauses during solving.

Queries

- Our primary focus lies on queries concerning clause utility, with the intention of examining the variables that exert a substantial influence on clause utility
- Our causal framework accommodates three distinct query categories, as detailed below:
- Average Treatment Effect (ATE) quantifies the variation in the outcome variable when the treatment variable shifts from one value to another.
- Conditional Average Treatment Effect (CATE) encapsulates the ATE, considering a condition for an additional variable.
- We propose the Averaged Conditional Average Treatment Effect (ACATE) concept to account for the need to average the CATE across all values of the conditional variable.
- (Averaged Conditional Average Treatment Effect). $\mathrm{ACATE}(X, Y, W, a, b) = \sum w(E[Y \mid do(X = a), W = w] - E[Y \mid do(X = b), W = w]) \Pr[W = w]$

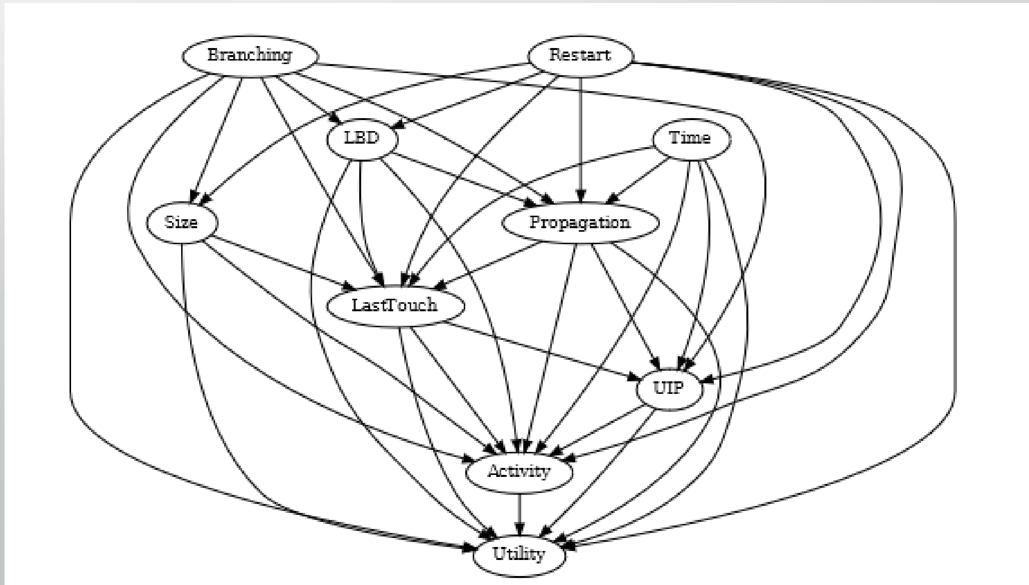


	Question	Query
Q1	Which clause, with low or high LBD, has greater utility?	$\mathrm{ATE}(\mathsf{LBD}, \mathsf{Utility}, 2, 1) < 0$
Q2	Which type of clause, large or small, has greater utility? What if the LBD is fixed?	$\begin{cases} \mathrm{ATE}(\mathsf{Size}, \mathsf{Utility}, 2, 1) < 0 \\ \mathrm{ACATE}(\mathsf{Size}, \mathsf{Utility}, \mathsf{LBD}, 2, 1) > 0 \end{cases}$
Q3	Which clause, with low or high LBD, experiences a rapid drop in utility over time?	$\begin{cases} \mathrm{CATE}(\mathsf{Time}, \mathsf{Utility}, \mathsf{LBD} \leq 6, 10000, 0) \geq 0 \\ \mathrm{CATE}(\mathsf{Time}, \mathsf{Utility}, \mathsf{LBD} > 6, 10000, 0) < 0 \end{cases}$
Q4	Which factor, size or LBD, has a greater impact on clause utility?	$\lvert\mathrm{ATE}(\mathsf{Size}, \mathsf{Utility}, 2, 1)\rvert > \lvert\mathrm{ATE}(\mathsf{LBD}, \mathsf{Utility}, 2, 1)\rvert$
Q5	Which factor, besides size, LBD, and activity, has the greatest impact on clause utility?	$\arg\max_{\mathsf{Treatment} \neq \mathsf{Utility}} \{\lvert\mathrm{ATE}(\mathsf{Treatment}, \mathsf{Utility}, 2, 1)\rvert\}$
Q6	Which branching heuristic, VSIDS or Maple, results in a greater clause utility?	$\mathrm{ATE}(\mathsf{Branching}, \mathsf{Utility}, \mathsf{Maple}, \mathsf{VSIDS})$
Q7	Which restart heuristic, Geometric, LBD-based, or Luby, results in the greatest clause utility?	$\begin{cases} \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathsf{Luby}, \mathsf{Geometric}) \\ \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathsf{Luby}, \mathsf{LBD\text{-}based}) \\ \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathsf{Geometric}, \mathsf{LBD\text{-}based}) \end{cases}$

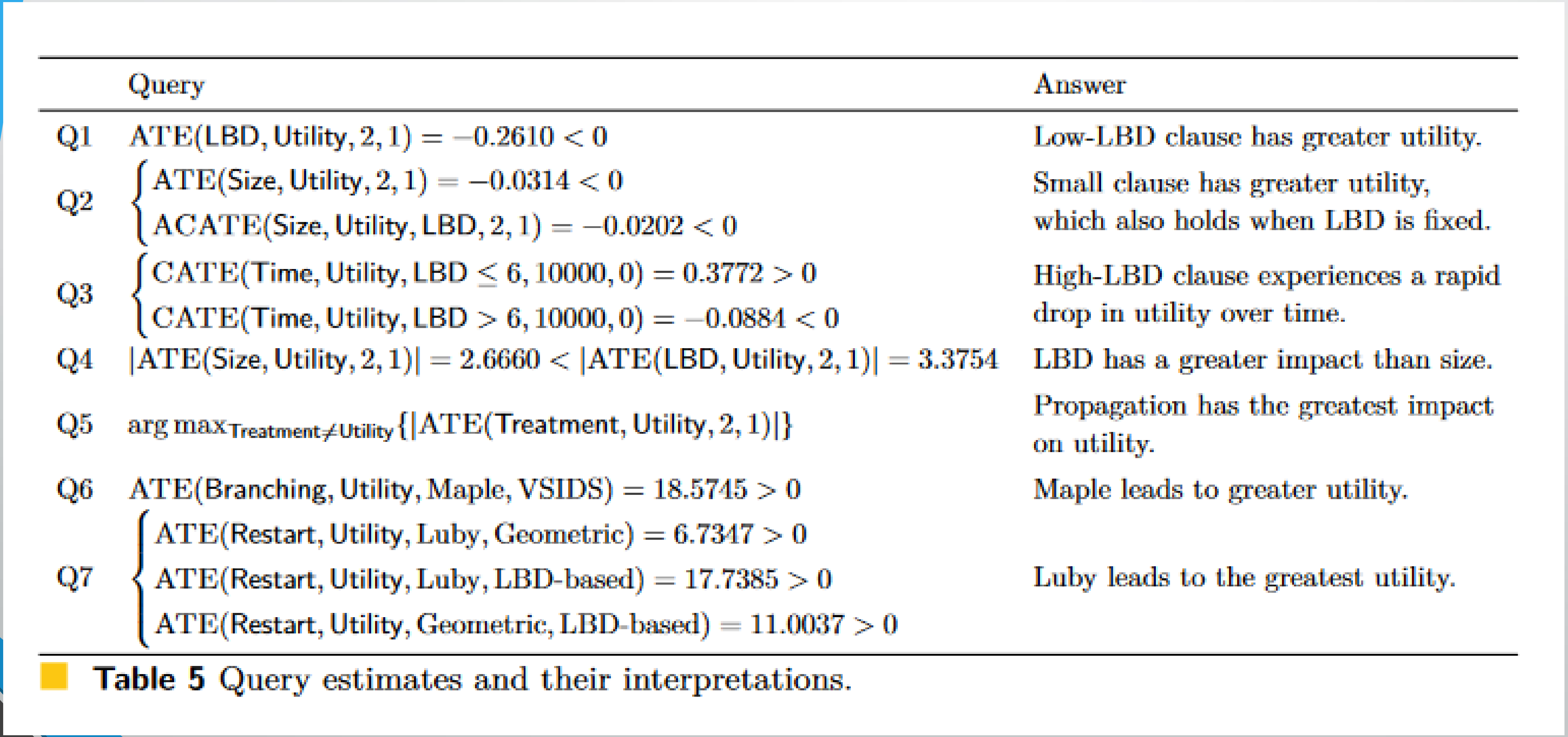
Table 2 Causal queries for questions on SAT solving.

Experiment Evaluation

- implement a prototype called CausalSAT to evaluate our approach over practical instances
- built on top of a structure-learning package, bnlearn, and a causal-reasoning package, DoWhy
- high-performance computing cluster
- could only use unsatisfiable instances
- use 80 UNSAT instances
- gathered 4 million data points
- an the hill-climbing algorithm using
- 10-fold cross-validation and obtained the causal graph



Query Answers



	Query	Answer
Q1	$\mathrm{ATE}(\mathsf{LBD}, \mathsf{Utility}, 2, 1) = -0.2610 < 0$	Low-LBD clause has greater utility.
Q2	$\begin{cases} \mathrm{ATE}(\mathsf{Size}, \mathsf{Utility}, 2, 1) = -0.0314 < 0 \\ \mathrm{ACATE}(\mathsf{Size}, \mathsf{Utility}, \mathsf{LBD}, 2, 1) = -0.0202 < 0 \end{cases}$	Small clause has greater utility, which also holds when LBD is fixed.
Q3	$\begin{cases} \mathrm{CATE}(\mathsf{Time}, \mathsf{Utility}, \mathsf{LBD} \leq 6, 10000, 0) = 0.3772 > 0 \\ \mathrm{CATE}(\mathsf{Time}, \mathsf{Utility}, \mathsf{LBD} > 6, 10000, 0) = -0.0884 < 0 \end{cases}$	High-LBD clause experiences a rapid drop in utility over time.
Q4	$\lvert\mathrm{ATE}(\mathsf{Size}, \mathsf{Utility}, 2, 1)\rvert = 2.6660 < \lvert\mathrm{ATE}(\mathsf{LBD}, \mathsf{Utility}, 2, 1)\rvert = 3.3754$	LBD has a greater impact than size.
Q5	$\arg\max_{\mathsf{Treatment} \neq \mathsf{Utility}} \{\lvert\mathrm{ATE}(\mathsf{Treatment}, \mathsf{Utility}, 2, 1)\rvert\}$	Propagation has the greatest impact on utility.
Q6	$\mathrm{ATE}(\mathsf{Branching}, \mathsf{Utility}, \mathrm{Maple}, \mathrm{VSIDS}) = 18.5745 > 0$	Maple leads to greater utility.
Q7	$\begin{cases} \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathrm{Luby}, \mathrm{Geometric}) = 6.7347 > 0 \\ \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathrm{Luby}, \mathrm{LBD\text{-}based}) = 17.7385 > 0 \\ \mathrm{ATE}(\mathsf{Restart}, \mathsf{Utility}, \mathrm{Geometric}, \mathrm{LBD\text{-}based}) = 11.0037 > 0 \end{cases}$	Luby leads to the greatest utility.

Table 5 Query estimates and their interpretations.

Conclusion

- first work to utilize causality to uncover the inner workings of modern SAT solvers
- proposed a framework to calculate the causal effects on clause utility from key factors such as LBD, size, activity, and the choice of heuristics
- the causal results verified the “rules of thumb” in solver implementation
- answered questions closely related to SAT solving
- causal framework provides a systematical way to quantitatively investigate the relationship among components of a SAT solver
- immediate focus will be enhancing our framework by introducing a new definition for clause utility
- frequency of a learned clause’s usage by a SAT solver
- aim to include a definition for clause utility that encompasses both satisfiable and unsatisfiable instances